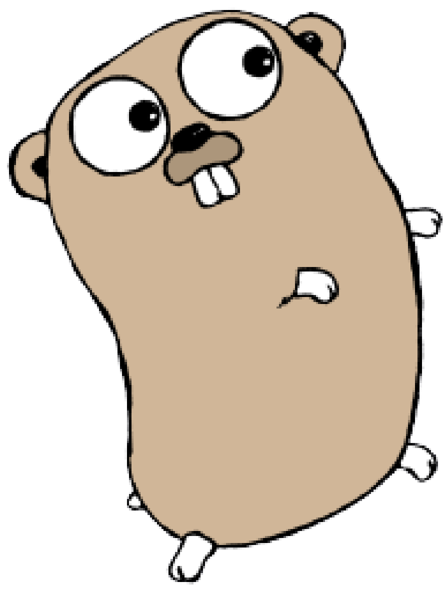
It's Go time!

Andreas Gustafsson & Camilla Romander

June 2, 2011



Abstract

The purpose of this report is to evaluate the functionality and syntax of Googles new programming language: Go!
To achieve this goal, several tests will be conducted. Trial-division, naive and optimal versions; The Sieve of Eratosthenes; Fibonacci generators, iterative, recursive and recursive with memorization; a small math-based game; and a wiki will be implemented. For comparison, each test will also be implemented in three other major languages: C, Java and Python. The powers of Go is its (allegedly) smooth syntax and fast runtime. It feels like an interpreted, dynamically typed language but is in fact a compiled, statically typed language.
Overall Go performed very well, it was faster than both Java and Python (even when not counting Javas extra second to boot the virtual machine) and in most cases as fast as C. In addition to the good runtime, it was easier to implement the tests in Go than in C and Java thanks to the simple syntax and orthogonality of the functions and packages.

Sammanfattning

Syftet bakom denna rapport är att utvärdera Googles nya programmeringsspråk, Go!
För att uppnå det målet ska flera tester implementeras. Dessa tester är naiva och optimala Fibonacci generatorer, Eratosthenes såll, naiva och optimala trial-division, ett litet matematikbaserat spel samt en wiki. För att få perspektiv på språket ska dessa tester dessutom implementeras i tre andra stora språk; Java, C och Python.
Gos styrkor är dess enkla syntax och snabba exekveringstid. Det känns som ett interpreterat, dynamiskt typat språk men är ett kompilerat, statiskt typat språk.
Överlag presterade Go väldigt bra, det var snabbare är både Java och Python (även om Javas en sekund extra tid för start av virtuell maskin inte räknas med) och i många fall lika snabbt som C, men det var väldigt mycket lättare att implementera saker i Go än i Java och C tack vare den enkla syntaxen och okomplicerade funktioner och paket.

Contents

List of Figures

Statement of collaboration

Task	**Performer**
Python implementations	Andreas
Java implementations	Camilla
Go implementations	Both
C implementations	Both
Web service design and implementation	Andreas
GUI design and implementation	Camilla
Writing the report	Both
Latex	Andreas
Analyzing results	Both

1 Introduction

1.1 About Go

Go is a fairly young programming language, developed by Google in these past few years[1] but it is only during the last year that its usage has become somewhat widespread. The language is designed by Robert Greisemer, Rob Pike and Ken Thompson who all are prominent characters within computer science. These three gentlemen sat down and discussed the big languages of today: Java, C, C++, C# and Python. They had the opinion that Java is too bloated, C is too meager, C++ is "just an extension of C to look more like Java", C# is "C++ for Windows" and Python is nice to code in, but quickly becomes messy. They quickly agreed that with their experience in computer science, they could combine these languages and learn from the mistakes of the past to make a new language that circumvents all the flaws in these languages but also encompasses the advances in hardware of recent years[2]. Thus Go was born. Go is a mix of all these languages, it has the quick compilation and runtime of C, the easy syntax of Python, a standard library similar to that of Java and C# and structs of C. Finally, they agreed that UNIX pipelines are really neat and so decided that UNIX–like pipelines will be used to allow threads to communicate without the extra code for synchronizing that is required in other languages[3].

1.2 Project description

The project is to test various abilities of the programming language Go and document the capabilities of this fairly young language. To perform these tests a Wiki was written in Go. This approach is perfect for the participants, since the first author is very interested in programming languages and find Go in particular very fascinating, and the second author is a User Interface Designer. This blend of interests is perfect for this project since these are the required skills to fully utilize the broad spectrum of features that is Go. Though a simple wiki is not a sufficient basis to evaluate an entire programming language on, a few other tests where also implemented and evaluated. These tests where Fibonacci generation (iterative, recursive and memorization), a small online game and prime number generation. Due to the limited time scope, the project will focus mainly, apart from the Wiki, on testing Go's capabilities regarding multi–threading applications, ease and smoothness of syntax and performance of runtime will be evaluated[4].

1.3 Approach

Since neither of us are well-versed in this language, the first few days was spent getting to grips with Go. We used a few online tutorials and the documentation on `http://golang.org` to produce several programs of varying sizes, from "Hello World!" to "Echo". This initial learning curve was quite steep, learning a new language might not seem like such a daunting task for an accomplished computer scientist but keep in mind that we only have

[1] The project was announced September 30th 2007
[2] Multi–core processors, powerful GPUs that can be used to compute arduous computations etc.
[3] http://golang.org
[4] According to `http://golang.org` the syntax is easy both to read and to write.

two or three languages below our belt. The time spent on simply learning the syntax is not reflected in this report. The GUI designer will prepare by reading "Don't make me think!" by Steve Krug and "Användarcentrerad Systemdesign" by Jan Gulliksen & Bengt Göransson and studying various wiki–pages such as wikipedia.com/en, wowwiki.com and go.wikia.com. The other participant will read online tutorials about multi–threading in several different languages to be better suited to compare with, and implement, multi–threading in Go. This mix of specialties is a good foundation for a wiki site about the performance of a particular programming language.
Included in this project is the work associated with learning an entirely new language. This is a difficult task in various degrees: the more experience you have, the easier it is to slip between different patterns of thought. Apart from simply writing a wiki, which can be accomplished in a few hours with code more or less stolen from a code-lab on `http://golang.org` we implemented several other programs, described in more detail later on in this report. Simply writing a wiki is utilizing too little of the langages potential to truly evaluate it.

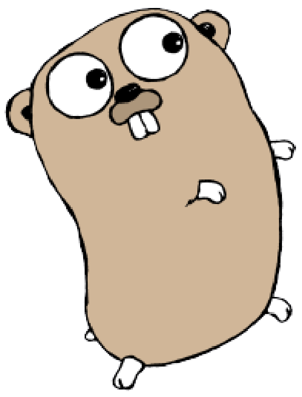


Figure 1: The gopher is a fitting mascot for Go, it is sleek and efficient.

2 Interview with Prof. Stefan Nilsson

How long have you programmed in Go?
Not very long, according to himself. He started in December, 2010.

How extensive is your knowledge about go?
He has written about 2000–3000 lines of Gocode and read the entire language specification and knows it well. He has not used each library package extensively, but he has a good overview of the various libraries.

What have you written in Go
All of the assignments in one of his courses at KTH, Introduction to Computer Science. Most of them were easy to implement, but a few of them, such as the graph oriented ones, required a complete redesign. This was necessary due to the lack of object orientation in Go. The largest project he has implemented in Go is a type of Bloomfilter with an API.

What are the syntactical benefits versus other languages?
There are many syntactical superiorities, none too revolutionary but all combined makes the syntax glorious. All the small annoying details of Java, C, C++ are fixed, such as semicolons and parentheses. There is very little he can complain about.

Are there any syntactical deficiencies?
The way the lack of semicolons is implemented is faulty, they are actually present, but it is optional to explicitly type them. This forces a special style of syntax. The parser (pre-compilation) appends a semicolon at the end of each line that *may* be an expression. This means you cannot start a line with a left curly bracket. ”There are a few amusing kinks in the syntax", as Prof. Nilsson himself expresses it.

Does Go demand a new way of reasoning when programming, and if so, what are your thoughts about it?
That all the packages are completely independent of each other. This decreases confusion and makes it easier to combine packages into a new function. If you are accustomed to hierarchal languages, you have to completely discard your way of thought and start over. The packages in Go are not hierarchal.

What are your opinion of the standard libraries in Go?
The design criteria of the libraries are that they are small, essential and relatively lightweight. If a module or function is ”nice to have sometime, maybe”, then it does not belong in the libraries. Only the absolute essentials belong there. This, as opposed to Javas libraries, makes it easy to get a good overview of the libraries.

Have you encountered any obstacles when writing Go code?
There are no libraries for GUIs, which forces you to use external libraries. A way to circumvent this flaw is, according to Prof. Nilsson, to write your application as a web service and let the browser be your GUI.

What is your opinion about channels v.s. synchronized objects?
This is an excellent idea. The concept is already explored in Erlang and used extensively in Unix in the form of pipes. Prof. Nilsson believes that channels are the solution to almost all concurrency and inter-application communication problems.

Tricks that are possible to do in Go that are impossible in other languages?
There are a few cool syntactical tricks that are possible in Go, but none that cannot be done in any other language. All programs written in Go can be written in all other languages with similar code. He thinks that Go is a language that is particularly enjoyable to code in and spend some of his spare time coding Go for the sheer joy of it.

3 Experiments

This section is a general description of each test to be conducted. A short introduction of each experiment will be presented, along with a description the aspects that are to be evaluated.

3.1 Computation of Fibonacci series

The Fibonacci sequence is an infinite sequence of integers defined by the following recurrence formula:

$$F_0 = 0, F_1 = 1, F_n = F_{n-2} + F_{n-1} \tag{1}$$

This is easy to implement in most programming languages and may be implemented by iteration or recursion. The most common implementation is using recursion with memorization[5]. This combines the power and easy–to–read syntax of recursion with storage of values in global parameters to achieve a small function that generates Fibonacci numbers.

In practice, Fibonacci series are used to compute the rate of expansion of a population given two seeds. The first seed is the number of individuals in a population, the second seed is the number of individuals *birthed* by the original population. Thus, based on this initial data, one can assume that the third number in the sequence, that is the current number of individuals, is the number of parents (the original herd) added to the newly spawned generation. Thus $Pop_{current} = Pop_{original} + Pop_{spawned}$ follows the definition of a Fibonacci series, but with slightly altered seeds.

In the following sections an analysis of the ease of implementation and performance of Fibonacci–generators in various languages.

3.2 Generation of Prime Numbers

Prime Numbers have a pivotal role within cryptography. The reasons for this are beyond the scope of this essay.
Generation of prime numbers consists of two steps; generating a number and checking if this number is a prime number. There are many different algorithms to produce these results, but most of them are fairly complex and are subjects for essays unto themselves. Therefore the focus of this essay was two of the simpler algorithms: trial–division and "The sieve of Eratosthenes". Both of these are techniques that are easy to grasp as well as being computationally demanding on the hardware.

Trial–division is a method for checking for primeness.

```
trialdiv(n):
    for int i in 2 to n-1:
        if n%i == 0:
            return not_prime
    return prime
```

[5] See code snippets in appendix B.1.

This is the naïve version. There are several optimizations, such as: if the number is not divisible by 2 it will not be divisible by any even number. Therefore it is only necessary to divide by the odd numbers in the range. Furthermore, some prime factor of n must be less than or equal to the square root of n.

```
trialdiv(n):
    int end = sqrt(n)
    if n%2==0: return false
    for int i in 3 to end by step 2:
        if n%i == 0:
            return false
    return true
```

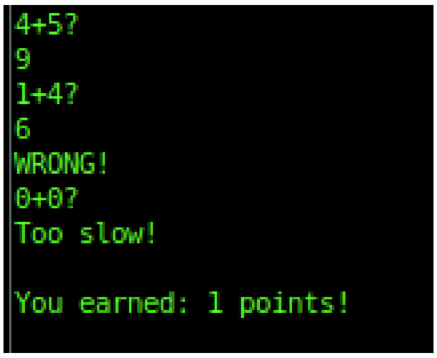
The sieve of Eratosthenes will generate all primes less than a given number. The algorithm is simple: Create a list from 2 to n, 2 is the smallest prime number, so strike out every other number in the list. Now we know that the next uncrossed number is a prime number, in this case three. Now strike out every third number from the list and proceed to the next uncrossed number 5. Now strike out every fifth number and proceed to 7. Keep doing this until you pass the sqrt(n)th element in the list. All remaining uncrossed numbers are primes.
In pseudo–code:

```
eratosthenes(n):
    list = [2,3,4,...,n]
    crossed = [false, false,..., false]
    for int i in 0 to n-1:
        if not crossed[i]:
            for int j in i to n-1 by step i:
                crossed[j] = true
    return list
```

3.3 Simple math-game

The game will simply print an easy mathematical problem (small sums to keep it simple) and the user shall supply the answer. One point is awarded for a correct answer and no points for an incorrect answer. The purpose of this test is to test the capabilities of concurrency i.e having a simplistic UI[6] updating multiple parts of the display at the same time. The program was be divided into two threads, one that receive input and one that use the received input and produces output. The same effect may be achieved by using non-blocking I/O, but that would defeat the purpose of the experiment. No effort at all was spent designing and implementing a fancy GUI[7], both because it is not in the scope of the test nor is it possible to do so in Go[8]. The client used one thread to read and send input, and one thread to receive and print output. The server randomized two numbers and stored the sum before sending the two numbers to the client. It then either award a point for the user if the answer was correct or it sends a new addition problem if the

[6]User Interface
[7]Graphical User Interface
[8]Go lacks any graphical libraries.



```
4+5?
9
1+4?
6
WRONG!
0+0?
Too slow!

You earned: 1 points!
```

Figure 2: Screenshot of mathgame in action.

user submitted an incorrect answer or the answer took more than five seconds to receive. All these processes ran concurrently and somewhat independently of each other. Fig. 2 illustrates how execution of the game is supposed to be.

4 The Wiki

4.1 Implementation

In the interview with Prof. Nilsson in section 2 it was revealed that there is no native support for GUIs[9] in Go. Therefore, it has been decided that all graphical components was to be written in HTML[10] or PHP[11]. These components was hosted on a Go web service. See B.4 for source code. Functionality of the wiki was mainly the viewing and editing of pages, but also inter page linking and navigation aid (a search bar or a list of pages). There are no URL to the finished wiki since this report will be eternally stored somewhere on KTH, we cannot publish a reliable URL. It is most likely that our server will go offline far before KTH will remove the report.

4.2 User Interface Design

The design was be heavily inspired by current popular wiki sites, such as `www.wikipedia.com/en` and `www.wowwiki.com`, since tools with the same purpose should have similar functionality according to Steven Krug. `www.wikipedia.com/en` already fulfills many of the criteria described in "Don't make me think!"; therefore many of the traits of the gowiki are inherited directly from the official wikipedia sites.

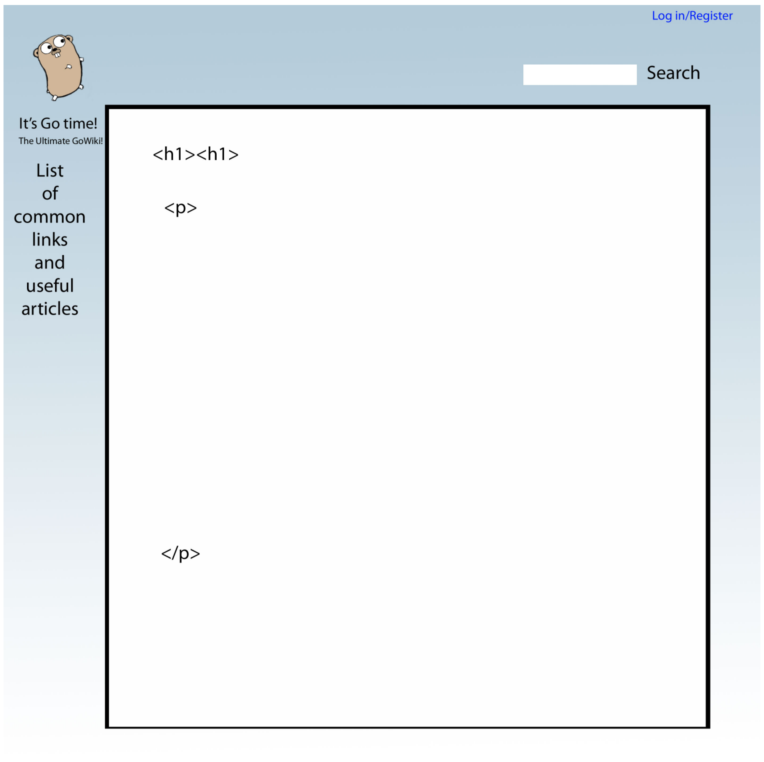


Figure 3: Initial draft for design layout of general page.

[9]Graphical User Interface
[10]Hyper Text Markup Language
[11]PHP: Hypertext Preprocessor

5 Results

5.1 Source of errors

The machine these tests were run on is not entirely stable. Due to Asus fancy energy hybrid software that reconfigured CPU-clock on the fly the results were not entirely trustworthy. Therefore the machine was tweaked with a clean installation of Ubuntu 10.4. Then the tests where run again under these new circumstances and new, more reliable results where had. All graphs are derived form data from the second round of measurements.

Another source of error is that all but the Java tests were run on the same machine. The Java tests were run on another machine, this contaminated the results.

This second round of tests each test was ran multiple times in a row, and the total time was considered. The important parts of these tests are not the absolute efficiency of the languages, by the relative efficiency.

5.2 Computation of Fibonacci series

5.3 A note about the recursive generator

The recursive Fibonacci generators have been omitted in this section since calculating any Fibonacci number greater than 45 takes a *very* long time. Calculating fib(50) = 12586269025 recursively, on the machine the tests were performed on, was not completed after fourteen hours. Computing fib(50) recursion 25172538049 recursion calls. Therefore any comparison of the languages based on the recursive generator is pointless, since the results are the same in all cases.

5.3.1 Java

The implementation of the Fibonacci generators was about 50 lines of code that are pretty straightforward. I/O is simple to use, the library Scanner was used to scan for integers. There where quite a few lines of codes that are "formalities", for instance the class definition and the main method signature are both tedious and tricky to write. All in all it was simple to implement. As can be seen in Fig. 4, Go was by far slower to complete its task than the other languages this result was startling since it was assumed that Java would be slowest. After some consideration though it was concluded that this slowness is due to the excessive output this program produces. The time for printing was accidentally included in this time, the actual time for calculating should be reminiscent of that of C.

5.3.2 C

C is almost like Java, around 50 lines of code. The main difficulties when implementing Fibonacci was handling the memorization part. All memory allocation in C must be handled manually. There was also some trouble with overflowing integers, it was quite problematic finding the correct structure (long long int) and finding the proper way to format these numbers into a string with printf.

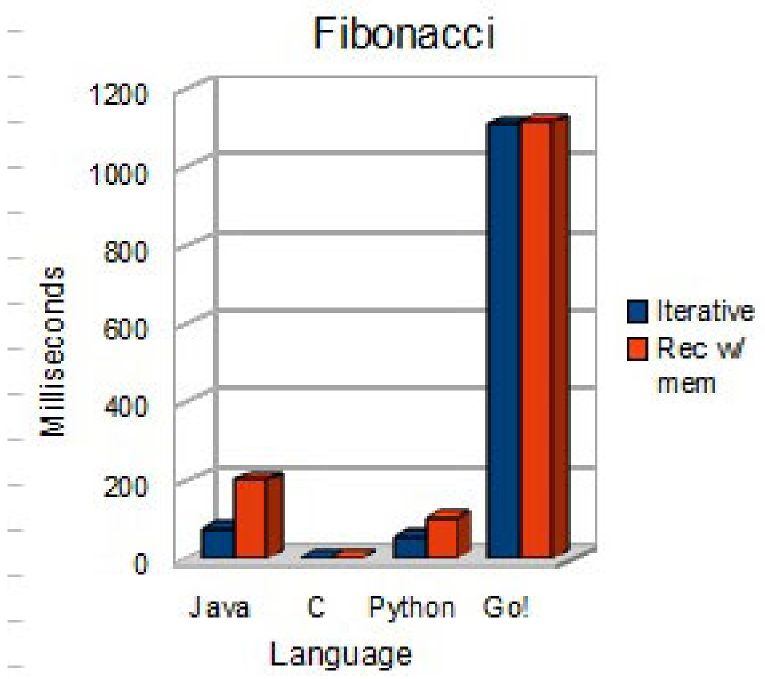


Figure 4: Graph of runtime when calculating the 90000th Fibonacci number 100 times. (see Appendix A for the number.

5.3.3 Python

The Python version was only 35 lines of code, and only 12 of them the actual generators. The rest of the code was a simple terminal based user interface and a few lines of code to time the individual functions. The most exciting part of this implementation was the use of direct-instantiation of a list of function pointers and then directly accessing one of them depending of the input. The array was never explicitly stored anywhere, and it allows a compact way to write simple flow-controlling code.

5.3.4 Go!

This is the implementation that required the most lines of code with 61 lines but many of them consists of a single right curly bracket, due to gofmt[12]. The syntax was at first glance similar to Java with all the curly brackets and indentation levels. Eventually one notices that there are no semicolons nor any explicit typing, and suddenly the code looks very pythonesque. Looking at the actual functions, they bear a significant reminiscence of C with names such as Printf and Scanf. Most of the difficulties encountered when implementing in Go was syntactical. Due to the programmers inexperience with the language (unlike their experience with the other languages) the code was complex. Part of the excessively long runtime may be die to the programmers unfamiliarity with this language, which makes them unable to perform the usual "performance hacks".

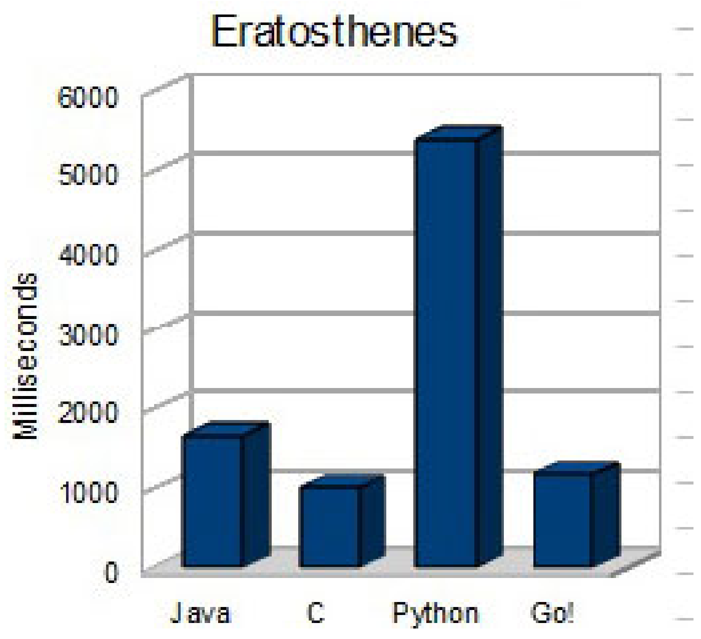


Figure 5: Graph of runtime when sieving forth the 900000 first Primes.

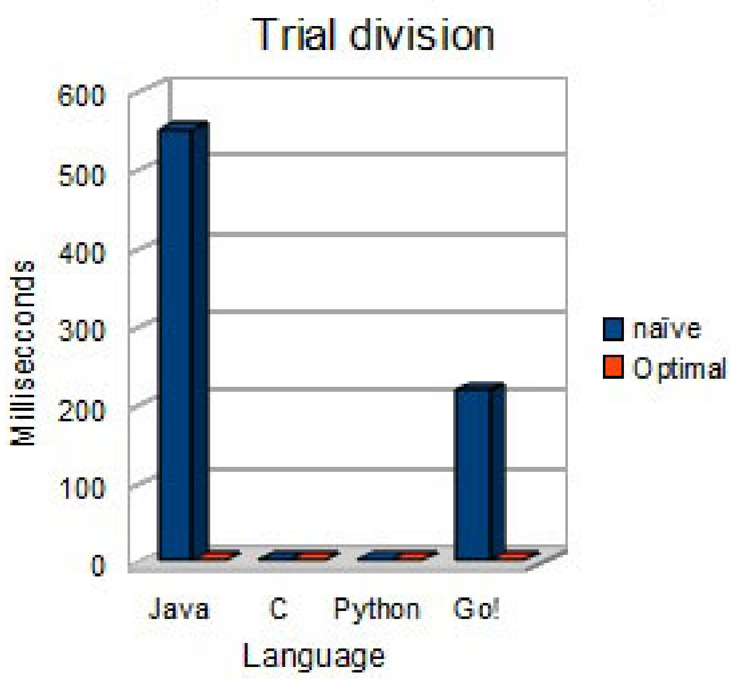


Figure 6: Graph of runtime when checking if 90001 is a prime by using trial-division.

5.4 Generation of Prime Numbers

5.4.1 Java

The implementation of The Sieve of Eratosthenes and Trial-division in Java was pretty much straightforward. I/O was still tedious with all the checked exceptions.
The overwhelmingly slow runtime of Java when executing Trial division was confounding at first

5.4.2 C

The Sieve of Eratosthenes implementation in C was particularly annoying because of arrays in C. The programmer must keep track of the length of each individual array, something that is not required in most other languages. Memory allocation is always hazardous and difficult to debug. A program written in C may compile and execute without crashing, but output or return values may be incorrect due to misaligned reads. This was the case during the sieving process when the array containing the primes was returned.

5.4.3 Python

As earlier, implementing anything small and script-like in python is simple and intuitive. Very few lines of code, the Sieve of Eratosthenes only required 26 lines of code. Seven of these where I/O and administrative code. The interesting part of the python implementation was the list comprehension. List comprehension does not exist in any of the other three languages.

5.4.4 Go!

This was where the most prominent syntactical flaw in Go was discovered. The equivalence of a long long int (from C) in Go is int64. When calculating the square root of an int64 one must first cast the int64 into a float64 since the math.Sqrt function only accepts a float64. Finally you must recast the return value from math.Sqrt from a float64 back to an int64. This sequence of casts may result in a loss of precision and makes the code less readable. The call described above would ultimately look like this:

```
var root int64 = int64(mat.Sqrt(float64(a)))
```

5.5 Math game

5.5.1 Java

Threading in Java is not very complicated, simply extend the Thread abstract class and override the run() method. Start the thread by invoking the start() method that is inherited from Thread. The start method will handle all administration required to spawn an additional process. The run method is basically equivalent to the main method, the only difference is that it is running in a disjoint process. Due to code conventions in Java, namely to place each class in a separate file, a process consisting of several smaller

[12] A formatting script that is built into the Go compiler.

threads quickly evolves into a project with many small files; some of which barely contain any code at all. See B.3.2 for examples.
Sockets on the other hand is a difficult, but important, concept to grasp and implement. There are many websites that attempt to explain sockets, but none that do so adequately. The use of sockets in Java is complicated further by inbuilt functions, such as proprietary I/O streams and functions. It is quite a daunting task for a relative "newbie" in the use of sockets to start to use them if you do it in Java. Other languages, such as Python, have much cleaner syntax for writing and reading from sockets. The checked exceptions in Java combined with all the possible errors generated by sockets (such as I/O error, host error, connection errors and so forth) makes the programs a mess of try/catch clauses.

5.5.2 C

Handling threads in C was not that difficult. Having them communicating with each other was the problem. Both threads received a pointer to a struct. This idiom is reminiscent of using a shared object. The problem was that, no matter how the semaphores where used, the threads kept modifying the data in the struct concurrently and with disastrous results. String handling in general in C is difficult without extensive knowledge about pointers and memory allocation. Conversions between integers and characters did not work until we remembered the ascii table. Sockets in C required a lot of code to initialize and much micromanaging when sending and receiving data. In general C was exhausting working with, after spending about twice the time on the game in C as was spent on Python, we simply gave up. The few minor bugs left was simply too frustrating and time-consuming to be bothered with. The bugs were all either string handling or concurrency issues.

5.5.3 Python

Implementing multithreaded applications in Python is theoretically simple. Just have your classes inherit from threading.Thread and override the run() method. To have two threads communicate with each other you create a class and give both threads the same instance of that class. No further synchronizing was needed (no mutexes or semaphores where used). The largest difficulty was synchronizing the shutdown, the sockets resisted closing and blocked the program. To solve this a global flag, encapsulated in the shared object, held a boolean. If that boolean became true during runtime, simply shut everything down.

5.5.4 Go!

Implementing multithreaded applications in Go is simple. Channels solve most synchronization and data sharing problems that have to be dealt with by synchronizing objects in other languages. No inheritance from a Thread superclass (as in Python and Java) is necessary, indeed it is even impossible since there are no hierarchal structure in Go. Simply write the keyword "go" in front of the method call to let it run concurrently. All methods and closures in Go have this ability.
Sockets behaved more unpredictably, but the methods in the "net" package are simple and the methods are clean. To establish a listening server only two lines of code are required. Even though the code was simple and easy to write, it did not behave as expected. The read functions read garbage data when they where not supposed to. This might be due

to lack of experience in this particular language. With a more comprehensive knowledge about Go and its runtime, these issues might be avoided with ease.

6 Conclusion

6.1 Go vs. Java

Not having to bother with semicolons and parentheses in loops and if statements is convenient. The number of lines of code required are about the same, but as a programmer become more experienced with a language the number of lines will decrease. The standard libraries in Java is much more extensive than those of Go, but Go have much less redundancy and old code in theirs. The libraries in Go are much cleaner than the libraries in Java, Java has become bloated over the years. Sharing information between threads in Java is meticulous and requires synchronized objects. Compared to that, the channels in Go is lightweight and easy to grasp for an inexperienced programmer.
Larger programs is easy to write in Java, mostly due to advanced and extremely useful IDEs such as NetBeans and Eclipse. These allow you to easily structure a class hierarchy and a catalogue-tree and gives you an excellent overview of the project as well as a powerful tool to utilize the libraries.
Go, due to its relative youth, lacks all these tools but as mentioned above the libraries are cleaner and more efficient than most of Javas.

Apart from pure syntactical advantages of Go over Java, the most striking benefit of Go is that go does not have checked exceptions. Java forces you to handle even the only remotely plausible errors. In Go, a function simply returns a value if something goes wrong, and the calling function may choose to ignore it. This makes the code more customizable and easy to read. Checked exceptions leads to frustration, frustration leads to poorly written error handling. Therefore the unchecked exceptions is a much better approach, let the programmer designer decide how to handle the errors.

6.2 Go vs. Python

Python is extremely easy to read and follow, but there are a few frustrating aspects of the syntax. Each access of a member-variable the programmer must explicitly type ”self.value”. This sort of referencing is tedious and makes the code less readable. Python have no curly brackets nor semi-colons, which makes the code look much cleaner. Go does have curly brackets which we personally dislike, but it does bring in the benefit of looking like more commonly used languages like C++ and Java. Syntactically and idiomatically Python and Go are very similar. Neither is explicitly typed, but only Python is dynamically typed. This allows Go-programmers to tweak their programs to a greater extent than Python programmers due to the optional ability to micromanage the typing.
Python is a slow language. It is not compiled so included in its runtime lies the interpretation. Go is exceptionally fast, in most regards as quick as C (but, as can be observed in the graphs, it has the potential to be very slow if you make mistakes). If performance is important, Go is your language. In any other case either one is fine as long as your application does not require a GUI[13]since Go lacks any and all support for those.

[13]Graphical User Interface

6.3 Go vs. C

All advantages of Go vs Java from 6.1 applies here as well. Apart from the syntax, the inbuilt memory management and garbage collection in Go is a huge plus. When coding in C we feel awkward and out of our comfort zone and there are a lot of statements and function calls that simply ”is required”. For instance the use of sockets in 5.5.2 required almost 30 lines of code to initialize the sockets and retrieve a connection. The same is achieved in Go with only three lines of code. C quickly becomes difficult to read and identify the important parts of the code and disregard the "formalities". Garbage collection exists in Go, it does not in C. This makes programming in Go easier since the allocation and freeing of memory is handled automatically and out of sight.

6.4 Performance

The performance of Go is close to that of C, it beats Java (even when not including the one second to boot the Java virtual machine) and Python by far. C is still the fastest language, but Go is a valiant second. Google claims that Go runs almost as fast as see, but this claim remains unproven. Though considering the ease of programming and much simpler syntax in Go than in C and the runner–ups few extra milliseconds of runtime in Go is, in our opinion, totally worth it.

7 Web services in Go

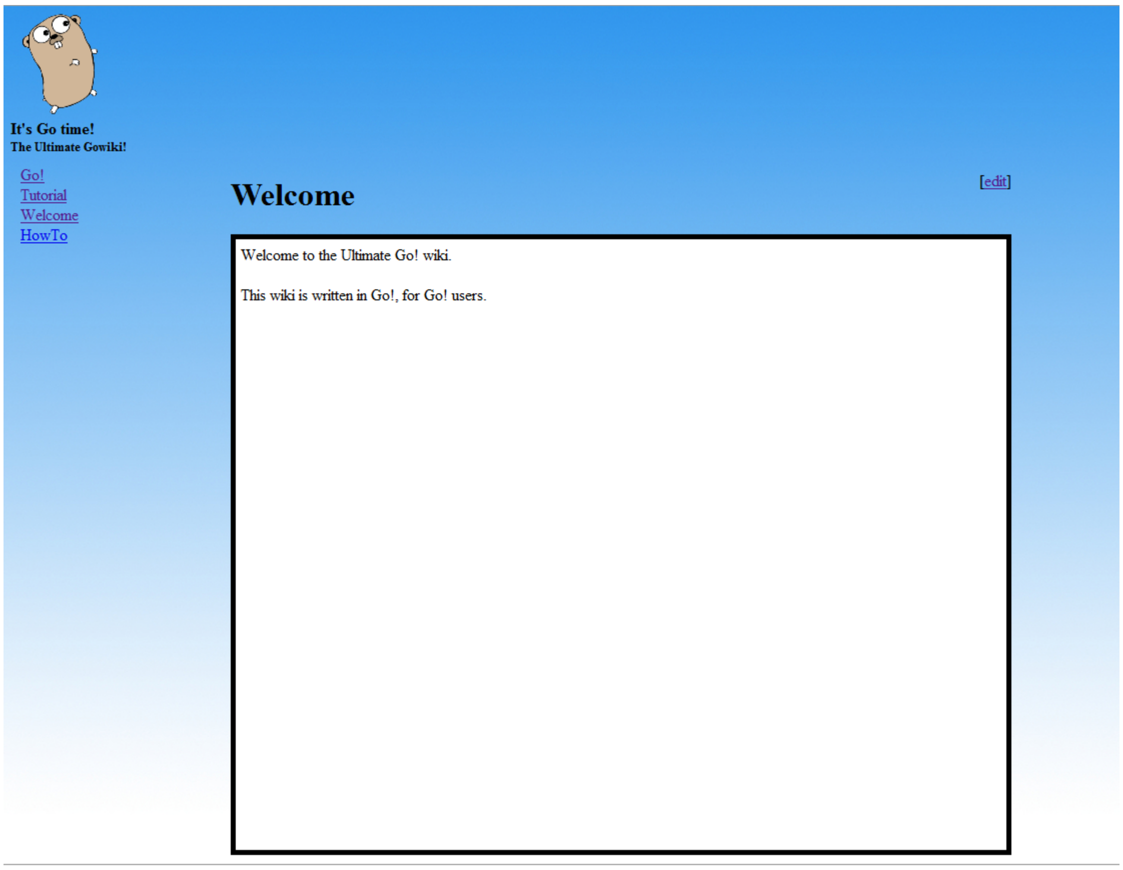
Writing a web service in Go is very simple, the ”net” package is simple to use and very powerful. Writing a wiki with a minimal interface, no error handling but with capabilities to edit and view pages is merely twenty lines of code. This code is easy to read and understand and simple to expand. Add error handling and the code ends up at about thirty lines of code that are still very readable. The only drawback is that there is no native support for a graphical user interface. The interface may be written in any language supported by a browser such as HTML, PHP or JavaScript (for this report HTML was chosen).

The functionality included in our web service is mainly viewing and editing wiki pages. Some small features added is the ability to hyper-link between wiki pages by writing the title of the page you wish to link to within square brackets. You may also hide which page you are linking to by specifying a display name for the link. Finally a list of available pages is constructed each time the page is updated since the lack of a search bar would make the site difficult to navigate. In Fig. 7 you may observe the final layout and design of the view page perspective and in Fig. 8 is the final layout and design of the edit page perspective.

7.0.1 Graphical Design of the Wiki

The design of the wiki has been kept simple to avoid confusion or promise advanced functionality that does not exist. It is immediately obvious what the purpose of the wiki is due to the extremely visible logo with the subscript ”The ultimate GoWiki!”. All links are colored blue (following the general design for links on the internet). Mimicking designs

from other pages with similar functions reduces the confusion of the user.
The purpose of the wiki page is to enlighten aspiring Go programmers, therefore all fancy (but ultimately unnecessary) features have been omitted. Due to lack of time, the creation of a search function was omitted, instead a list of links to all pages were added to the left side bar.



Welcome

Welcome to the Ultimate Go! wiki.

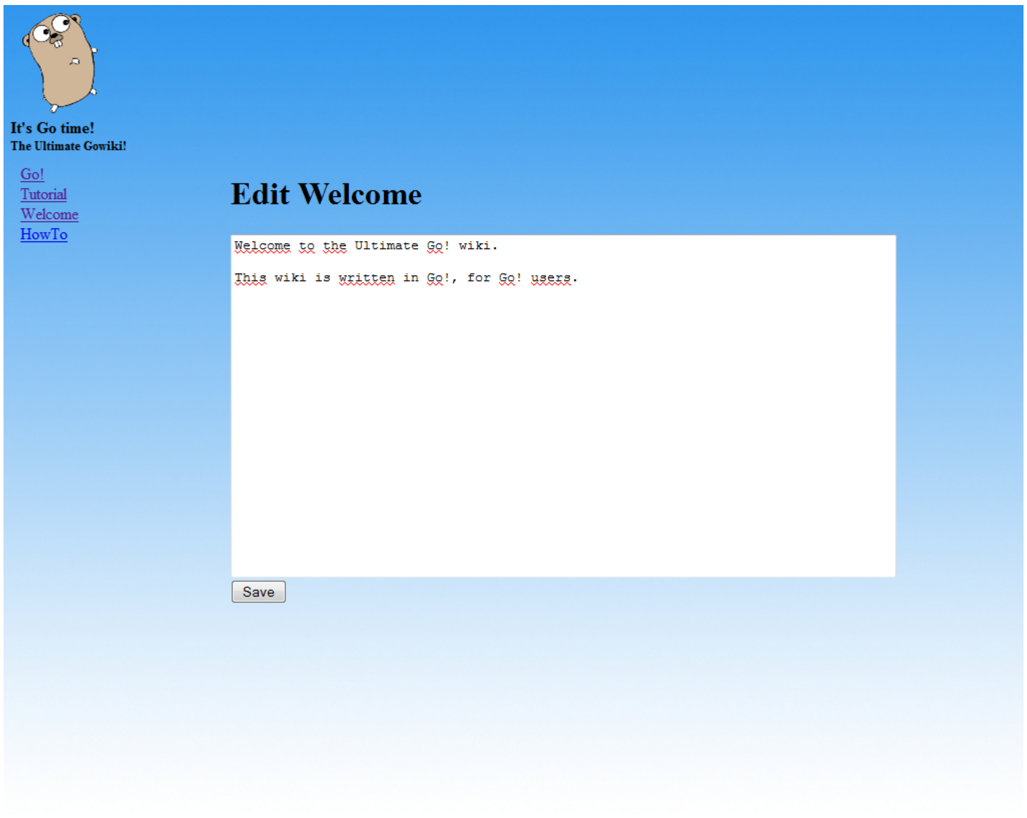
This wiki is written in Go!, for Go! users.

Figure 7: The final design of the view page perspective.

8 Final Evaluation

Overall the language is very nice, though there are a few flaws. One of those flaws is a very strict compiler. For instance, one may not import a package or declare a variable and then end up not using it. This is apparently a compile error, but in most other languages it would be a warning at most. This makes it difficult to plan your code. Assume that you know that the packages that will be used are "fmt", "os" and "net" and you connect two sockets and verify that this is successful by compiling and trying. The compiler will now give you an error because you have imported but not used the "os" package yet. We believe a better solution would be to give a flag to the compiler if you are in "development mode" or want to compile a release version. Very similar to the -O flag of the common C compilers.
The compiler[14] generates very efficient code and compiles swiftly. Contrary to Python

[14] 6g,8g. Not gomake.



Edit Welcome

```
Welcome to the Ultimate Go! wiki.

This wiki is written in Go!, for Go! users.
```

Figure 8: The final design of the edit page perspective.

that evaluates slowly but gets the job done without too much difficulties.
A very nice touch is that exceptions are unchecked. A method may have an additional return value that may be an error. Then it is up to the caller to decide whether to care or just ignore the error. This decreases frustration and makes the code cleaner. Javas checked exceptions leads to lots of auto-generated code and sloppy error handling. For instance, an I/O unit which make sure that input is always an integer, but returns a string (this is common in C#). Then converting this string into an integer will never result in a NumberFormatException, but Java will still force you to handle it.

One of the biggest flaws is that there are no packages to handle GUIs. This makes Go a language that is difficult to use in commercial cases, since almost all applications requires some sort of User Interface. If this flaw persists, Go will become a language used only on the server side. While being well suited for that role we believe that Go can evolve into a strong competitor to the C family.

Writing web services is very easy and well integrated into the language. Since an increasing portion of communication transfer onto the internet, the logical conclusion is that a new language should fit into these new systems with ease. Go has achieved this goal well, the difference between "Hello World!" and "Hello World!, the web service edition" is less than four lines of code. Most other languages quickly become more complex when communicating through the internet.

Go is still a young language and has plenty of time to evolve. These flaws might be resolved in the future, and if so, Go might be, along with C, C++, C# and Java, one of the Major Languages.

The opinions expressed by Prof. Nilsson in section 2 matches well with the above assessments. His extensive experience as a programmer gives weight to our relatively inexperienced insights into the language of Go!

The more languages you know, the easier it is to learn another one.

9 Litterature

This is a list of litterature read prior to commencing the project.

- Don't make me think!, Steve Krug
- Användarcentrerad Systemdesign, Jan Gulliksen & Bengt Göransson
- www.golang.org/documentation, 2011-04-10

A Fib(9000)

fib(9000) = 34616029128668474631328927294065319582100493884057464919779235488262676145124

B Source Code

B.1 Computation of Fibonacci series

B.1.1 Python

```python
#!/usr/bin/python

from time import time
def iterative(n):
    a, b = 0, 1
    for x in xrange(n):
        a,b=b,a+b
    return a
blargh = 0
def recursive(n):
    blargh = blargh + 1
    if n < 2: return n
    return recursive(n-2) + recursive(n-1)

#Dictionary<Integer,<Integer>
mem = {0:0,1:1}
def recursive_mem(n):
    if n not in mem: mem[n] =
        recursive_mem(n-1)+recursive_mem(n-2)
    return mem[n]

if __name__ == "__main__":
    #Main loop
    number = input("Number: ")
    while True:
        print("1. Iterative.\n2. Recursive.\n3. Recursive with
            memorisation.")
        k = input(">")
        #This is a trick one may use to work around the lack of
            switch cases in python.
        #You place delegate functions in a list and immediately
            pick which one by using the index-operator.
        try:
            start = time()
            for x in range(100):
                    [iterative, recursive,
                        recursive_mem][k-1](number)
                end = time()
                print("Time: %dms."%(end-start))
            except IndexError:
                print "Invalid input."
                continue
```

B.1.2 Java

```
import java.util.Scanner;

/**
 * Will find the fibaonacci number of your choice.
 * @author Camilla Romander
 *
 */
public class Fibonacci1 {

    public static void main(String[] args){
        int n = chooseNumber();
        long t = System.currentTimeMillis();
        if(n == -1){
            return;
        }
        long n0 = 0;
        long n1 = 1;
        long n2 = 0;
        for(int k=0; k<100; k++)
        {


            for(long i = 0; i < n-1; i++){
                n2 = n0 + n1;
                n0 = n1;
                n1 = n2;

            }
        }
            System.out.println(" " + n1);
            System.out.println("Time:  " +
               (System.currentTimeMillis() - t)+ " ms");
    }


    /**
     * Will decide if your number is positive.
```

```java
     * @return
     */
    public static int chooseNumber(){
        System.out.println("Enter your number and it can not be
           negative");

        Scanner sc = new Scanner(System.in);
        int i = sc.nextInt();
        if (i >= 0){
            System.out.println("You numbers is " + i);
            return i;
        }else
            System.out.println("Your number is not a positive
               number");
        return -1;

    }

}
```

```java
import java.util.Scanner;

/**
 * Will find the fibonacci number of your choice in a recursive
    way.
 * @author Camilla Romander
 *
 */

public class FibonacciR {

    public static void main(String[] args){
        int n = chooseNumber();
        long t = System.currentTimeMillis();
        System.out.println(rec(n));
        System.out.println("Time:  " +
           (System.currentTimeMillis() - t)+ " ms");
    }

    public static int rec(int n){

        if(n<2){
            return n;
        }else
            return rec(n-2) + rec(n-1);

    }

    /**
     * Will decide if your number is positive.
     * @return
     */
    public static int chooseNumber(){
        System.out.println("Enter your number and it can not be
           negative");

        Scanner sc = new Scanner(System.in);
        int i = sc.nextInt();
        if (i >= 0){
            System.out.println("You numbers is " + i);
            return i;
        }else
            System.out.println("Your number is not a positive
               number");
        System.exit(1);
        return 0;

    }


}
```

```java
import java.util.Scanner;

/**
 * Will find the fibonacci number of your choice and will do it
    fast.
 * @author Camilla Romander
 *
 */
public class FibonacciRM {

    static int[] fib;
    public static void main(String[] args){
        int k = chooseNumber();
        long t = System.currentTimeMillis();
        fib = new int[k+1];
        fib[0] = 0;
        fib[1] = 1;
        for(int u=0; u<100;u++){
        for(int i = 2; i < k+1; i++){
            fib[i] = -1;
        }
        }
        System.out.println(recM(k));
        System.out.println("Time:   " +
           (System.currentTimeMillis() - t)+ " ms");

    }

    public static int recM(int n){

        if(fib[n] == -1){
            fib[n] = recM(n-2) + recM(n-1);

        }return fib[n];


    }

    /**
     * Will decide if your number is positive.
     * @return
     */
    public static int chooseNumber(){
        System.out.println("Enter your number and it can not be
           negative");

        Scanner sc = new Scanner(System.in);
        int i = sc.nextInt();
        if (i >= 0){
            System.out.println("You numbers is " + i);
            return i;
        }else
            System.out.println("Your number is not a positive
               number");
        System.exit(1);
        return -1;

    }

}
```

B.1.3 C

```
#include <stdio.h>
#include <string.h>
#include <time.h>

long long iterative(long long n)
{
    long long a = 0, b = 1, i, c;
    for(i=0;i<n-1;i++)
    {
        c = a+b;
        a = b;
        b = c;
    }
    return b;
}

long long rec(long long n)
{
    if(n<2)
        return n;
    return rec(n-2) + rec(n-1);
}

long long rec_mem(long long n, int *values)
{
    if(values[n] == -1)
        values[n] = rec_mem(n-2, values) + rec_mem(n-1, values);
    return values[n];
}

int main(int argc, char **argv)
{
    long long choice, number, res;
    while(1)
    {
        printf("1. Iterative\n2. Recursive\n3. Recursive with
           memorization.\n");
        scanf("%lld%lld",&choice, &number);
        int values[number+1];
        long start = time(0);
        int i;
        for(i=0;i<100;i++){
        switch(choice)
        {
            case 1: res = iterative(number); continue;
            case 2: res = rec(number); continue;
            case 3:
                memset(values,-1,(number+1)*sizeof(int));
                   values[0] = 0; values[1] = 1;
                res = rec_mem(number, values);
                continue;
        }
        }
        printf("fib(%lld)=%lld\n",number, res);
        printf("Time: %ldms\n",(time(0)-start));
    }
}
```

B.1.4 Go!

```
package main

import "fmt"
import "time"
func iterative(n int) int64 {
    var a int64 = 0
    var b int64 = 1
    for i := 0; i < n-1; i++ {
        var c int64 = a + b
        a = b
        b = c
    }
    return b
}

func recursive(n int64) int64 {
    if n < 2 {
        return n
    } else {
        return recursive(n-2) + recursive(n-1)
    }
    return 0
}

func rec_mem(n int64, calced []int64) int64 {

    if calced[n] == -1{
        calced[n] = rec_mem(n-2,calced)+rec_mem(n-1,calced)
    }
    return calced[n]
}

func main() {
    n := 0
    var res int64
    for {
        res = 0
        fmt.Printf("1. Iterative.\n2. Recursive\n3. Recursive
           with memorization\n")
        fmt.Scanf("%d%d", &res, &n)
        start := time.Nanoseconds()
        for i:=0;i<100;i++{
        if res == 1 {
            res = iterative(n)
        } else {
            if res == 2 {
                res = recursive(int64(n))
            } else {
                //slice allocation
                s := make([]int64, n+1)
                // range = enumerate from python
                for i,_ := range s {
                    s[i] = -1
                }
                s[0] = 0
                s[1] = 1
                res = rec_mem(int64(n),s)
            }
        }
        }
        fmt.Printf("fib(%d) = %d\n", n, res)
        fmt.Printf("Time: %dms.\n",
           (time.Nanoseconds()-start)/1000000)
    }
}
```

B.2 Generation of Prime Numbers

B.2.1 Python

```
#!/usr/bin/python
from time import time

def trial_division_naive(n):
    for x in xrange(2,n+1):
        if n%x==0: return False
    return True

from math import sqrt
def trial_division_opt(n):
    if n%2==0: return False
    k = int(sqrt(n))
    for x in xrange(3,k+1,2):
        if n%x==0: return False
    return True

```

```
if __name__ == "__main__":
    while True:
        choice = input("1. Naive trial division\n2. Optimal trial division.\n")-1
        start = time()
        for x in range(10):
            div = [trial_division_naive, trial_division_opt][choice]
        print "Prime! :D" if div(input("Number: ")) else "Is not prime :("
        print("Time: %dms."%(time()-start))
```

```
#!/usr/bin/python
from math import sqrt
from time import time

def eratosthenes(k):
    nums = range(2,k+1)
    #If crossed[i] is True, then nums[i] is not a prime.
    crossed = [False for x in nums]
    breakpoint = int(sqrt(k)) + 1
    #i = current index, x = nums[i]
    for i,x in enumerate(nums):
        if crossed[i]: continue
        if x > breakpoint: break
        #Eg: 2 is the current number, then we wish to cross 4, 6, 8, 10...
        for y in xrange(x+i,len(nums),x): crossed[y] = True
    #return a generator expression for all primes less than k
    return (x for i,x in enumerate(nums) if not crossed[i])

if __name__ == "__main__":
        start = time()
        p = 900000
        for x in range(10):
            k = eratosthenes(p)
        end = time()
        for x in k:
          print x,
        print("\nTime: %dms."%(end-start))
```

B.2.2 Java

```
import java.util.Scanner;


public class Eratosthenes {

    static int[] primeNumbers;
    static boolean[] list;

    public static void main(String[] args){

    int n = chooseNumber();
    int breakN = (int)Math.sqrt(n);
    primeNumbers = new int[n];
    list = new boolean[n];
    long t = System.currentTimeMillis();
    for(int k=0; k<10; k++)
    {
    if(n == 0){
            return;
       }else

    for(int i = 2; i < n; i++){

         primeNumbers[i] = i;
         list[i] = false;
    }
    for(int i = 2; i < breakN+1; i++){
         if(list[i] == true){
             continue;
         }
       for(int y = primeNumbers[i]+i; y < primeNumbers.length;
          y += primeNumbers[i]){

            list[y] = true;
       }

    }
    }
    for (int i = 2; i < primeNumbers.length; i++) {
         if(list[i] == false){
             System.out.print(primeNumbers[i]+ " ");
         }

   }
    System.out.println("Time:  " + (System.currentTimeMillis()
       - t) + " ms");
}



```

```
    /**
     * Will decide if you number is bigger then zero.
     * @return
     */
    public static int chooseNumber(){
        System.out.println("Enter your number and it has to be
           larger then 1");

        Scanner sc = new Scanner(System.in);
        int i = sc.nextInt();
        if (i > 1){
            System.out.println("You numbers is " + i);
            return i;
        }else
            System.out.println("Your number is not larger than
               1");
        return 0;

    }
}
```

```
import java.io.BufferedReader;
import java.util.Scanner;


/**
 * Can decide if the number your have entered is a prime number.
 * @author Camilla Romander
 *
 */
public class PrimeNumbersNaive {
    static BufferedReader Input;

    public static void main(String[] args){
        int n = chooseNumber();
        long t = System.currentTimeMillis();
        if(n == 0){
            return;
        }else

        for(int i = 0; i < 10; i++){
        for ( int Index = 2; Index < n; Index++ ) {

            if ( n % Index == 0 ) {

                System.out.println ( n + " is not a prime
                   number." );
```

```
                    System.out.println("Time:   " +
                       (System.currentTimeMillis() - t) + " ms");
                    return;
                }
            }
        }
        System.out.println ( n + " is a prime number." );
        System.out.println("Time:   " +
           (System.currentTimeMillis() - t) + " ms");
    }


    /**
     * Will decide if you number is bigger then zero.
     * @return
     */
    public static int chooseNumber(){
        System.out.println("Enter your number and it has to be
           larger then 0");

        Scanner sc = new Scanner(System.in);
        int i = sc.nextInt();
        if (i > 0){
            System.out.println("You numbers is " + i);
            return i;
        }else
            System.out.println("Your number is not larger than
               0");
        return 0;

    }

}
```

```
import java.util.Scanner;

/**
 * Can decide if the number your have entered is a prime number.
 * @author Camilla Romander
 *
 */
public class PrimeNumbersOptimal {

    public static void main(String[] args){

        int n = chooseNumber();
        long t = System.currentTimeMillis();
        if(n == 0){
            return;
        }else

        for(int i=0; i<10; i++){
        if(n % 2 == 0){

            System.out.println("This was not a prime number.");
            System.out.println("Time:  " +
               (System.currentTimeMillis() - t) + " ms");
        }
        else{
        for ( int Index = 3; Index <= (int)Math.sqrt(n); Index
           += 2 ) {

            if ( n % Index == 0 ) {

                System.out.println ( n + " is not a prime
                   number." );
                System.out.println("Time:  " +
                   (System.currentTimeMillis() - t) + " ms");
                return;

            }
        }
        }
        }
        System.out.println ( n + " is a prime number." );
        System.out.println("Time:  " +
           (System.currentTimeMillis() - t) + " ms");
    }
    /**
     * Will decide if you number is bigger then zero.
     * @return
     */
    public static int chooseNumber(){
        System.out.println("Enter your number and it has to be
           larger then 0");

        Scanner sc = new Scanner(System.in);
        int i = sc.nextInt();
        if (i > 0){
            System.out.println("You numbers is " + i);
            return i;
        }else
```

```
                System.out.println("Your number is not larger than
                    0");
            return 0;

        }

}
```

B.2.3 C

```
#include <stdio.h>
#include <math.h>
#include <time.h>

int trialdiv_naive(int n)
{
    int i;
    for(i=2;i<n;i++)
        if(n%i==0) return 0;
    return n<2?0:1;
}

int trialdiv_opt(int n)
{
    int i,br = (int) sqrt(n);
    if(n%2 ==0)
        return 0;
    for(i =3; i<br;i+=2)
        if(n%i == 0) return 0;
    return 1;
}

int main(int argc, char **argv)
{
    while(1)
    {
        printf("1. Naive.\n2. Optimal.\n");
        int choice,number,res;
        scanf("%i%i",&choice,&number);
        time_t start = time(0);
        int i;
        for(i=0;i<10;i++)
        {
            switch(choice)
            {
                case 1: res = trialdiv_naive(number); continue;
                case 2: res = trialdiv_opt(number); continue;
            }
        }
        //Formatting output wether the number is a prime or not
            using a ternary statement.
        printf("%i is %s\n",number, !res?"not prime :(":"
            prime! :D");
        printf("Time: %lims.\n",(time(0)-start)*1000);
    }
    return 1;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

int eratosthenes(int n, int *nums)
{
    int i,j,num_primes = 0, br = (int)sqrt(n);
    for(i=2;i<=br;i++)
    {
        if(nums[i])
        {
            continue;
        }
        for(j=2*i;j<n;j+=i)
        {
            nums[j] = 1;
        }
    }
    for(i=2;i<n;i++)
    {
        if(!nums[i])
        {
            nums[num_primes++] = i;
        }
    }
    return num_primes;
}
int main(int argc, char **argv)
{
    int n, i, *primes;
    while(1)
    {
        printf("Number: ");
        scanf("%i",&n);
        time_t start = time(0);
        //allocation of result array, calloc initializes the
           memory to 0
        primes = (int*) calloc(n,sizeof(int));
        int i, number_primes = 0;
        for(i = 0; i<10; i++)
            number_primes = eratosthenes(n,primes);
        for(i=0;i<number_primes;i++)
        {
            printf("%i ", primes[i]);
        }
        printf("\n");
        free(primes);
        printf("Time: %lims.\n",(time(0)-start)*1000);
    }
    return 1;
}
```

B.2.4 Go!

```
package main

import "fmt"
import "math"
import "time"

func trialdivNaive(n int64) bool {
    var i int64
    for i = 2; i < n; i++ {
        if n%i == 0 {
            return false
        }
    }
    return true
}


func trialdivOpt(n int64) bool {
    var i int64
    br := int64(math.Sqrt(float64(n)))
    if n%2 == 0 {
        return false
    }
    for i = 3; i < br; i += 2 {
        if n%i == 0 {
            return false
        }
    }
```

```go
    return true
}

func main() {
    var n, choice int64
    var isPrime bool
    for {
        fmt.Print("1. Naive\n2. Opt\n")
        fmt.Scanf("%d%d", &choice, &n)
        start := time.Nanoseconds()
        for i:=0; i < 10; i += 1{
        if choice == 1 {
            isPrime = trialdivNaive(n)
        } else {
            isPrime = trialdivOpt(n)
        }
        }
        if isPrime {
            fmt.Printf("%d is prime! :D\n", n)
        } else {
            fmt.Printf("%d is not prime :(\n", n)
        }
        fmt.Printf("Time: %dms.\n",
           (time.Nanoseconds()-start)/1000000)
    }
}
```

```go
package main

import "fmt"
import "math"
import "time"

func eratosthenes(n int, primes []int) int{
    br := int(math.Sqrt(float64(n)))
    numPrimes := 0
    for i:=2;i<=br;i++{
        if primes[i] == 1{
            continue
        }
        for j:=2*i;j<=n;j+=i{
            primes[j] = 1
        }
    }
    for i:=2;i<n;i++{
        if primes[i]==0{
            primes[numPrimes] = i
```

```
                numPrimes++
            }
        }
        return numPrimes
}

func main(){
        var n int
            n = 900000
            start := time.Nanoseconds()
            primes := make([]int,n+1)
            for i:=0;i<n+1;i++{
                primes[i]=0
            }
            var numPrimes int
            for i:=0; i<10; i++{
            numPrimes = eratosthenes(n,primes)
            }
            for i:=0;i<numPrimes;i++{
                fmt.Printf("%d ",primes[i])
            }
            fmt.Printf("\nTime:
               %dms.\n",(time.Nanoseconds()-start)/10000000)
}
```

B.3 Math-game

B.3.1 Python

```
#!/usr/bin/python
import threading
import socket
import sys
import time
import random
import datetime

class inp(threading.Thread):
    """Receives input from the client and makes it visible for
       the sender thread."""
    def __init__(self, sock):
        threading.Thread.__init__(self)
        self.sock = sock
        self.valid = False
        self.mess = 0
        self.shutdown = False
    def run(self):
        while True:
            try:
                if self.valid:
                    time.sleep(0.1)
                    continue
                if self.shutdown:
                    self.sock.close()
                    return
                self.mess = int(self.sock.recv(256))
                self.valid = True
            except ValueError:
                shutdown = True

class sender(threading.Thread):
    """Receives input from the inp thread and sends output to
       the client.

    If the client has not answered within 5 seconds, send a
       "wrong" message and a new assignment.
    If the client has answered within 5 seconds and the answer
       is correct, award the user with 1 point and send a new
       assignment.
    If the client has answered within 5 seconds and the answer
       is incorrect, send a "wrong" message and a new
       assignment.
    If three assignments has been sent, send the number of
       points to the client and terminate. """
    def __init__(self, sock):
        threading.Thread.__init__(self)
        self.sock = sock
        self.inp = inp(sock)
        self.currentAnswer = 0
        self.currentPoints = 0
        self.timeOut = 50
        self.timeCount = 0
        self.rnd = random.Random()
        self.sendQ()
    def run(self):
        self.inp.start()
        c = 1
        while True:
            if self.inp.shutdown:
                self.sock.close()
                return
            if self.timeCount < self.timeOut and not
               self.inp.valid:
                time.sleep(0.1)
                self.timeCount += 1
                continue
            if self.inp.valid and self.inp.mess == self.currentAnswer:
                self.currentPoints += 1
            else:
                if self.timeCount >= self.timeOut:
                    self.sock.send("Too slow!")
                else:
                    self.sock.send("WRONG!")
            if c == 3:
                break
            self.inp.valid = False
            self.sendQ()
            c += 1
        self.sock.send("\nYou earned: %i points!\n"%self.currentPoints)
        self.inp.shutdown = True
    def sendQ(self):
        a = self.rnd.randint(0,5)
        b = self.rnd.randint(0,5)
        self.currentAnswer = a+b
        self.timeCount = 0
        self.sock.send("%s+%s? "%("(%i)"%a if a<0 else "%i"%a ,"(%i)"%b if b<0 else "%i"%b))
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(('', int(sys.argv[1])))
server.listen(1)

#Listen for players forever and start a game for each connection.
while True:
    sock, details = server.accept()
    print(details)
    sender(sock).start()
```

```
import sys
import socket
import threading

die = False
class recv(threading.Thread):
    """Receive input from the server and print it to the user."""
    def __init__(self, sock):
        threading.Thread.__init__(self)
        self.sock = sock
    def run(self):
        while True:
            if die:
                return
            inp = sock.recv(256)
            print(inp)

class send(threading.Thread):
    """Receive input from the user and send it to the server."""
    def __init__(self, sock):
        threading.Thread.__init__(self)
        self.sock = sock
    def run(self):
        while True:
            inp = raw_input()
            if inp == "q":
                die = True
                sock.close()
                return
            sock.send(inp)

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect(("localhost", int(sys.argv[1])))
recv(sock).start()
send(sock).start()
```

B.3.2 Java

```
package client;

import java.io.IOException;
import java.net.Socket;
import java.net.UnknownHostException;

public class Client {

    public static void main(String[] args) {
        try {
            Socket s = new Socket("localhost", 8080);
            new CReceiver(s).start();
            new CSend(s).start();
        } catch (UnknownHostException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
```

```java
                e.printStackTrace();
            }
    }
}
```

```java
package client;

public class ClientObj {
    String mess = null;
    boolean valid = false;
    boolean shutDown = false;


}
```

```java
package client;

import java.io.IOException;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;

public class CReceiver extends Thread {
    Scanner sc;
    Socket s;
    PrintWriter pw;
    public CReceiver(Socket s){
        this.s = s;
        sc = new Scanner(System.in);
        try {
            pw = new PrintWriter(s.getOutputStream());
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }
    public void run(){
        while(true){
            String i = sc.nextLine();
            pw.println(i);
        }
    }
}
```

```java
package client;

```

```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.Socket;

public class CSend extends Thread {
    BufferedReader bf;
    Socket s;

    public CSend(Socket s){
        this.s = s;
        try {
            bf = new BufferedReader(new
                InputStreamReader(s.getInputStream()));
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    public void run(){
        while(true)
        try {
            System.out.println(bf.readLine());
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

}
```

```java
package server;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;


public class Quiz {
    ServerSocket server;
    public Quiz(){
        try {
            server = new ServerSocket(8080);
        } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

    }
    public static void main(String[] args) {
        new Quiz().dolit();

    }

    private void dolit() {
        ServerOb ob = new ServerOb();
        Socket client = null;
        PrintWriter out = null;
        BufferedReader in = null;
        try {
            client = server.accept();
        } catch (IOException e) {
            System.out.println("Accept failed: 4444");
            System.exit(-1);
        }
        try {
            out = new PrintWriter(
                        client.getOutputStream(), true);
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        try {
            in = new BufferedReader(
                        new InputStreamReader(
                            client.getInputStream()));
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        new Receiver(in, client, ob).start();
        new Sender(client, ob, out).start();

    }
}
```

```
package server;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.net.Socket;


public class Receiver extends Thread {

    private final BufferedReader in;
    private final Socket s;
    private final ServerOb ob;
    public Receiver(BufferedReader in, Socket s, ServerOb ob){
        super();
        this.in = in;
        this.s = s;
        this.ob = ob;

    }
    public void run(){
        while(true){
            if(ob.shutDown == true){
                try {
                    in.close();
                    s.close();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
            if(ob.valid == true){
                try {
                    Thread.sleep(100);
                    continue;
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }else{
                try {
                    ob.mess = Integer.parseInt(in.readLine());
                    ob.valid = true;
                } catch (NumberFormatException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }

            }
        }
    }


}
```

```
package server;

import java.io.PrintWriter;
import java.net.Socket;
import java.util.Random;


public class Sender extends Thread {
    private final ServerOb ob;
    private final Socket s;
    private PrintWriter pw;
    private Random r;

    public Sender(Socket s, ServerOb ob, PrintWriter pw){
        super();
        r = new Random();
        this.ob = ob;
        this.s = s;
        this.pw = pw;
    }
    public void run(){
        int time = 0;
        int numberOfQuestion = 1;
        int answer = questions();
        int numberOfPoints = 0;
        while(true){
            if(ob.shutDown == true){
                pw.close();
                return;
            }
            if(ob.valid == false || time == 50){
                try {
                    Thread.sleep(100);
                    time++;
                    continue;
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }else if(numberOfQuestion == 3){
```

```java
                    pw.println("You got " + numberOfPoints +
                        "number of points.");
                    break;
                }if(ob.mess == answer){
                    numberOfPoints++;

                }else{
                    pw.println("Wrong answer stupid!!!");
                }
                answer = questions();
                numberOfQuestion++;


            }
        }
        public int questions(){
            int number1 = r.nextInt(5);
            int number2 = r.nextInt(5);
            String question = (" " + number1 + " + " + number2+
                "?");
            pw.println(question);
            int answer = number1 + number2;
            return answer;
        }

}
```

```java
package server;

public class ServerOb {
    int mess = 0;
    boolean valid = false;
    boolean shutDown = false;
}
```

B.3.3 C

```c
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <time.h>

```

```

typedef struct pack
{
    int valid;
    char mess;
    int shutdown;
    int sock;
}shared_t;

void error(const char *msg)
{
    perror(msg);
    exit(1);
}


void initrand()
{
    srand((unsigned)(time(0)));
}

int randint()
{
    return rand();
}

//generates a psuedo-random integer between 0 and max
int randint_m(int max)
{
    return (int)max*rand()/(RAND_MAX+1.0);
}

//generates a psuedo-random integer between min and max
int randint_r(int min, int max)
{
    if (min>max)
    {
      return max+(int)(min-max+1)*rand()/(RAND_MAX+1.0);
    }
    else
    {
      return min+(int)(max-min+1)*rand()/(RAND_MAX+1.0);
    }
}

char send_q(int newsockfd)
```

```
{
    char a = (char)randint_r(0,5);
    usleep(1000);
    char b = (char)randint_r(0,5);
   //char a = 'a', b = 'b';
    char snd[6] = {a+48,'+',b+48,'?',' ', '\0'};
    write(newsockfd,snd,6);
    return (char)(a+b+48);
}

void sender(void *arg)
{

      int time_out = 50, time_count = 0, points = 0, c = 1;
      initrand();
      shared_t *shared = (shared_t*)arg;
      char answer = send_q(shared->sock);
      while(1)
      {
        if(!shared->valid && time_count < time_out)
        {
            usleep(100000);
            time_count++;
            continue;
        }
        else{
            if(shared->mess == answer)
                points++;
            else
                write(shared->sock, "WRONG!", 6);
            if(c==3)
                break;
           answer = send_q(shared->sock);
           time_count = 0;
           c++;
//             write(shared->sock,"\nSetting valid to false\n",
   24);
           shared->valid = 0;
        }
      }
      char snd[21] = "You earned:   points!";
      snd[12] = (char)(points+48);
      write(shared->sock,snd,21);
      shared->shutdown = 1;
      pthread_exit(0);
}
```

```

void receiver(void *arg)
{
    shared_t *shared = (shared_t*)arg;
    while(1)
    {
        if(shared->valid)
        {
            usleep(100000);
            continue;
        }
        if(shared->shutdown)
        {
            close(shared->sock);
            pthread_exit(0);
        }
        int res = read(shared->sock,&shared->mess,1);
        if(res != 1)
            continue;
       // shared->mess+=48;
        write(shared->sock,"\nSetting valid to true.\n", 25);
        shared->valid = 1;
    }
}

int main(int argc, char *argv[])
{
     int sockfd, newsockfd, portno;
     socklen_t clilen;
     struct sockaddr_in serv_addr, cli_addr;
     if (argc < 2) {
         fprintf(stderr,"ERROR, no port provided\n");
         exit(1);
     }
     sockfd = socket(AF_INET, SOCK_STREAM, 0);
     if (sockfd < 0)
        error("ERROR opening socket");
     bzero((char *) &serv_addr, sizeof(serv_addr));
     portno = atoi(argv[1]);
     serv_addr.sin_family = AF_INET;
     serv_addr.sin_addr.s_addr = INADDR_ANY;
     serv_addr.sin_port = htons(portno);
     if (bind(sockfd, (struct sockaddr *) &serv_addr,
              sizeof(serv_addr)) < 0)
              error("ERROR on binding");
     listen(sockfd,5);
     clilen = sizeof(cli_addr);
     newsockfd = accept(sockfd,
                 (struct sockaddr *) &cli_addr,
                 &clilen);
     if (newsockfd < 0)
          error("ERROR on accept");

    pthread_t send, recv;
    shared_t args = {0, ' ', 0, newsockfd};
    pthread_create(&send, NULL, sender, ((void*)(&args)));
    pthread_create(&recv, NULL, receiver, ((void*)(&args)));

   pthread_join(send,NULL);
   pthread_join(recv,NULL);
   close(sockfd);
   return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <pthread.h>

void error(const char *msg)
{
    perror(msg);
    exit(0);
}
int sockfd;
void sender(void* arg)
{
    char in;
    while(1)
    {
        scanf("%c",&in);
        write(sockfd, &in,1);
    }
}

void receiver(void* arg)
{
    char in[30];
    while(1)
    {
        bzero(in,30);
        read(sockfd,in,30);
        printf("%s",in);
    }
}


int main(int argc, char *argv[])
{
    int portno, n;
    struct sockaddr_in serv_addr;
    struct hostent *server;

    char buffer[256];
    if (argc < 3) {
       fprintf(stderr,"usage %s hostname port\n", argv[0]);
       exit(0);
    }
    portno = atoi(argv[2]);
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");
    server = gethostbyname(argv[1]);
    if (server == NULL) {
        fprintf(stderr,"ERROR, no such host\n");
        exit(0);
    }
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    bcopy((char *)server->h_addr,
         (char *)&serv_addr.sin_addr.s_addr,
         server->h_length);
    serv_addr.sin_port = htons(portno);
    if (connect(sockfd,(struct sockaddr *)
       &serv_addr,sizeof(serv_addr)) < 0)
        error("ERROR connecting");

    pthread_t send, recv;
    pthread_create(&send,NULL,sender,(void*)0);
    pthread_create(&recv,NULL,receiver,(void*)0);
    //pthread_join(send,NULL);
    //pthread_join(recv,NULL);
    return 0;

```

```
}
```

B.3.4 Go!

```
package main

import "fmt"
import "net"
import "rand"
import "time"
import "math"

func receiver(fd net.Conn, c chan int) {
    defer fd.Close()
    for {
        b := make([]byte, 1)
        _, err := fd.Read(b)
        if err != nil {
            c <- -1
            return
        }
        c <- int(b[0] + 48)
    }
}

func sender(fd net.Conn, c chan int) {
    defer fd.Close()
    answer := sendQ(fd)
    points := 0
    timeOut := 50
    timeCount := 0
    numberOfQ := 1
    for {
        if numberOfQ == 3 {
            break
        }
        select {
        case in := <-c:
            if in == -1 {
                return
            }
            if in == answer {
                points++
            } else {
                b := []byte("WRONG! ")
                b[5] = uint8(in)
                fd.Write(b)
                }
                answer = sendQ(fd)
                numberOfQ++
                timeCount = 0
            default:
                if timeCount < timeOut {
                    time.Sleep(int64(math.Pow10(8)))
                    timeCount++
                    continue
                }
                fd.Write([]byte("Too slow!"))
                timeCount = 0
                numberOfQ++
                answer = sendQ(fd)
            }
        }
        b := []byte("You earned   points!")
        b[11] = uint8(points+48)
        fd.Write(b)
}

func sendQ(fd net.Conn) (answer int) {
    n1 := uint8(rand.Int31n(6) + 48)
    n2 := uint8(rand.Int31n(6) + 48)
    b := []byte(" + ?")
    b[0] = n1
    b[2] = n2
    fd.Write(b)
    answer = int(n1 + n2)
    return
}

func main() {
    server, err := net.Listen("tcp", "127.0.0.1:8080")
    if err != nil {
        fmt.Println("ERROR1!")
        return
    }
    for {
        fd, _ := server.Accept()
        fmt.Println("Recvd conn!")
        c := make(chan int)
        go receiver(fd, c)
        go sender(fd, c)
    }
}
```

```go
package main

import "fmt"
import "net"


func recv(fd net.Conn, c chan int){
    buf := make([]byte, 1)
    defer fd.Close()
    for{
        select{
        case i1 := <-c:
                if i1 == -1{
                    return
                }
         default:
            fmt.Scanln(buf)
            fd.Write(buf)
        }
    }
}


func send(fd net.Conn, c chan int){
    b := make([]byte, 256)
    defer fd.Close()
    for {
      r, err := fd.Read(b)
            if err !=nil{
                c<--1
                return
            }
        fmt.Println(string(b[:r]))
        fmt.Printf("%d\n",r)
   }
}

/*
  Create a connection to the server and start the two
     goroutines.
   Wait for them to terminate.
*/
func main(){
    c ,err := net.Dial("tcp", "", "127.0.0.1:8080")
    if err != nil{
        fmt.Println("ERROR! :(")
        return
    }
    defer c.Close()
    ch := make(chan int)
    go recv(c, ch)
    go send(c, ch)
    select{
    case i1 := <-ch:
            ch <- i1
            return
    }
    return
}
```

B.4 The Wiki

B.4.1 Go

```
package main

import (
    "http"
    "os"
    "io/ioutil"
    "template"
    "regexp"
    "strings"
    "fmt"
)

type Page struct {
    Title string
    Body []byte
    Menu []byte
}


const lenPath = len("/view/")

var internalLinkingHidden =
   regexp.MustCompile("\\[[a-zA-Z0-9]+\\|[a-zA-Z0-9]+\\]")
var internalLinking = regexp.MustCompile("\\[[a-zA-Z0-9]+\\]")
var newLines = regexp.MustCompile("\n")
func viewHandler(w http.ResponseWriter, r *http.Request, title
   string) {
    p, err := loadPage(title)
    if err != nil{
        http.Redirect(w,r, "/edit/"+title, http.StatusFound)
        return
    }
    p.Body = newLines.ReplaceAllFunc(p.Body, func(item []byte) []byte {return []byte("<br />")})
    p.Body = internalLinkingHidden.ReplaceAllFunc(p.Body, replaceHiddenLinks)
    p.Body = internalLinking.ReplaceAllFunc(p.Body, replaceLinks)
    renderTemplate(w, "view", p)
}

func listOfLinks() []byte{
    datadir, err := os.Open("data", os.O_RDONLY, 0666)
    if err != nil{
        fmt.Println("FACKA UUUUUUUUUUUUR!")
    }
    conts, _ := datadir.Readdir(-1)
    ret := ""
    var pageName string
    for _, x := range conts{
        pageName =  strings.Split(x.Name,".", 2)[0]
        ret += "<a href=/view/"+pageName+">"+pageName+"</a><br /br>"
    }
    return []byte(ret)
}

func replaceLinks(item []byte) []byte{
    return []byte("<a href=/view/"+string(item[1:len(item)-1])+">"+string(item[1:len(iter
}

func replaceHiddenLinks(item []byte) []byte{
    s := strings.Split(string(item), "|", 2)
    return []byte("<a href=/view/"+s[1][:len(s[1])-1]+">"+s[0][1:]+"</a>")
}

func saveHandler(w http.ResponseWriter, r *http.Request, title string){
    body := r.FormValue("body")
    p := &Page{Title: title, Body: []byte(body)}
    err := p.save()
    if err != nil{
        http.Error(w, err.String(), http.StatusInternalServerError)
        return
    }
    http.Redirect(w, r, "/view/"+title, http.StatusFound)
}

func editHandler(w http.ResponseWriter, r *http.Request, title string) {
    p, err := loadPage(title)
    if err != nil {
        p = &Page{Title: title}
    }
    renderTemplate(w, "edit", p)
}

func makeHandler(fn func (http.ResponseWriter, *http.Request, string)) http.HandlerFunc{
    return func(w http.ResponseWriter, r *http.Request){
        title := r.URL.Path[lenPath:]
        if !titleValidator.MatchString(title){
            http.NotFound(w,r)
            return
        }
        fn(w,r,title)
    }
}

func repl(item []byte) []byte {
    return []byte("<a href=/view/"+string(item[1:len(item)-1])+">"+string(item[1:len(item)-
}

func renderTemplate(w http.ResponseWriter, tmpl string, p *Page) {
    p.Menu = listOfLinks()
//  err := templates[tmpl].Execute(w, p) //uncomment this line to enable template caching
    err := template.MustParseFile("tmpl/"+tmpl+".html",nil).Execute(w, p) //comment this line to enable template chaching
    if err != nil{
        http.Error(w, err.String(), http.StatusInternalServerError)
    }
}
```

```

func loadPage(title string) (*Page, os.Error) {
    filename := "data/"+title + ".txt"
    body, err := ioutil.ReadFile(filename)
    if err != nil {
        return nil, err
    }
    return &Page{Title: title, Body: body}, nil
}

func (p *Page) save() os.Error {
    filename := "data/"+p.Title + ".txt"
    return ioutil.WriteFile(filename, p.Body, 0600)
}

func getTitle(w http.ResponseWriter, r *http.Request) (title
   string, err os.Error){
    title = r.URL.Path[lenPath:]
    if !titleValidator.MatchString(title){
        http.NotFound(w,r)
        err = os.NewError("Invalid page title")
    }
    return
}

var titleValidator = regexp.MustCompile("^[a-zA-Z0-9]+$")
var templates = make(map[string]*template.Template)
func init(){

    for _,tmpl := range []string{"edit", "view"}{
        templates[tmpl] =
           template.MustParseFile("tmpl/"+tmpl+".html",nil)
    }
}


func main() {
    http.HandleFunc("/view/", makeHandler(viewHandler))
    http.HandleFunc("/edit/", makeHandler(editHandler))
    http.HandleFunc("/save/", makeHandler(saveHandler))
    http.HandleFunc("/", func (w http.ResponseWriter, r
       *http.Request){http.Redirect(w,r,"/view/Welcome",
       http.StatusFound)})
    http.ListenAndServe(":8081", nil)
}
```

B.4.2 HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
    <head>
        <meta http-equiv="Content-type" content="text/html;
           charset=utf-8" />
    </head>
<body style = "background:
   url('http://www.suksamaipile.com/image/BlueToWhiteGradient.jpg');
   background-repeat:repeat-x;">

<div style = "width: 1000px;">
<img
   src="http://news.cnet.com/i/bto/20091109/go_gopher_color_logo_250x249.png"
   width = "100" height = "100"/>
    <h4 style = "margin: 0px; padding: 0px;"> It's Go time!</h4>
    <h5 style = "margin: 0px; padding: 0px;"> The Ultimate
       Gowiki!</h5>

</div>


<div style="margin: 0px; width: 1000px; height: 1000px">


<div style="float: left; margin: 10px; width: 200px; height:
   100%; background-color:clear">
        {Menu}
</div>

<div style="float:right; width: 780px">
<div style="margin: 0px; height: 60px;">
    <div style="float:left;">
    <h1>{Title}</h1>
    </div>
                <div style = "float: right;">
                <p>[<a href="/edit/{Title}">edit</a>]</p>
</div>
<div style = "clear:both;">
</div>
</div>

<div style="margin: 0px; height: 600px; background-color:white;
   border-style: solid; border-color: black; border-width:
    5px;clear:both;padding:5px;">

    {Body}
</div>
</div>
</div>
</div>
</body>
</html>
```

```






<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
    <head>
        <meta http-equiv="Content-type" content="text/html;
            charset=utf-8" />
    </head>
<body style = "background:
    url('http://www.suksamaipile.com/image/BlueToWhiteGradient.jpg');
    background-repeat:repeat-x;">


<div style = "width: 1000px;">
<div>
<img
    src="http://news.cnet.com/i/bto/20091109/go_gopher_color_logo_250x249.
    width = "100" height = "100"/>
<h4 style = "margin: 0px; padding: 0px;"> It's Go time!</h4>
<h5 style = "margin: 0px; padding: 0px;"> The Ultimate
    Gowiki!</h5>

</div>

<div style="margin: 0px; width: 1000px; height: 1000px">
<div style="float: left; margin: 10px; width: 200px; height:
    100%; background-color: clear">
        {Menu}
</div>

<div style="float:right; width: 780px">
<div style="margin: 0px; height: 60px;">
```

```
    <div style="float:left;">
    <h1>Edit {Title}</h1>
    </div>
<div style = "clear:both;">
</div>
</div>
                    <form action="/save/{Title}" method="POST">
                    <div><textarea name="body" rows="20"
                       cols="80">{Body|html}</textarea></div>
                    <div><input type="submit"
                       value="Save"></div>
                    </form>



    </div>
</div>
</div>
</div>
</body>
</html>
```