CPSC-354 Report

Michael Masakayan
Chapman University

September 12, 2022

Abstract

Short summary of purpose and content.

1 Introduction

1 My name is Michael Masakayan, I'm a 4th year computer science major. I transferred from Pasadena City College in 2020. I am interested in cyber security and although I won't be able to get a minor in it I plan on getting a few certificates this year. I enjoy playing video games and like watching movies

2 Homework

This section will contain your solutions to homework.

2.1 Week 1

Assignment 1: GCD. The program figures out the Greatest Common Divisor for any given two numbers using Euclid's algorithm.

1. Firstly you want to put the program into your IDE and run g++ -o assignment1 assignment.cpp
2. then type in ./assignment1.cpp
3. Once you run the program you type in the first number then you hit enter then you type in the second number and hit enter. It will then show you the GCD of the two numbers.

I chose to use c++ because I am familiar with the language and it was easier for me to write the program in it. This assignment was mainly for us to get used to using LaTeX

```
#include <iostream>
using namespace std;
int gcd(int n, int m) {
   if (m == 0)
   {
       return n;
   }

   else if(n>m)
   {
      gcd(m,n-m);
   }
   else if(n<m)
```

```
   {
      gcd(n,m-n);
   }
   return gcd(m, n % m);
}
int main() {
    int n,m;
    cout<<"type the number for the first argument of the GCD, then hit enter"<<'\n';
    cin>>n;
    cout<<"type the number for the first argument of the GCD,then hit enter"<<'\n';
    cin>>m;
   cout<<"GCD of "<< n <<" and "<< m <<" is "<< gcd(n, m);
   return 0;
}
```

2.2 Week 2

Assignment 2: In this assignment we were tasked with creating a few recursiver functions in haskel that is

1. select$_e vens select_o dds$
2. member
3. append
4. revert
5. less$_e qual$

Select evens will take a list and return a another list with only the even elements of the list in the argument. select$_o dds is meant to do what select even does but for the odds items in the list. Member, is meant to check a list if it has the given argu$

6. Firstly you want to put the program into your IDE and run ghci
7. then type in :load assignment2.hs
8. you can type the following and they should give the results
   (a) select$_e vens$["e","f","g","h","i"] $--result$["f","h"]

(b) select$_o dds$["e","f","g","h","i"] $--result$["f","h","i"]
(c) member 3 [2,4,3]–result true member 3 [2,4,2]–result false
(d) append [2,1] [3,1,2] –result [2,3,1,1,2]
(e) revert [5,6,7,8] –result [8,7,6,5]
(f) less$_e qual$[1][2] $--result true less_e qual$[3][2] $--result false$
I mainly had issue with less equal. I could not find the syntax for how to recursively go through more than one element lists. Most of them used the same type of recursion.

```
{-let ezample_eveny = ["a","b","c","d","e"] -}
{-yelect_eveny (z:zy) n = z !!n if z !! n/2 == % 0 then b:a!! n
-}
select_evens:: [a] -> [a]
select_evens [] = []
select_evens [x] = []
select_evens (x1:x2:xy) = x2:select_evens xy
-- selects all the odd elements in a list
```

```
select_odds :: [a] -> [a]
select_odds [] = []
select_odds [x] = [x]
select_odds (x1:x2:xy) = x2:select_odds xy
-- member takes an int and a list and checks if the int is in the list

member:: Eq a=> a-> [a] -> Bool
member _ [] = False
member x (z:zs)= (x `elem` zs) == True



-- combines two lists into one list
append :: [a] -> [a] -> [a]
append [] xy = xy
append (z:zy) xy = z:append xy zy
-- reverses the order of a list
revert :: [a] -> [a]
revert xs = rev [] xs where
  rev :: [a] -> [a] -> [a]
  rev acc   []  =      acc
  rev acc (x:xs) = rev (x:acc) xs
--compares the values of the lists
less_equal:: Ord a=> [a] -> [a] -> Bool
less_equal [][] = True
less_equal [x][b] = (x>=b)
-- less_equal (x:xs) (b:bz) = x<b:less_equal xs bz



main::IO ()
main = print(append [1,2] [3,4,5])
-- print(less_equal [][])
-- print(select_odds ["a","b","c","d","e"])
-- print (revert [1,2,3])
```

...

3 Project

Introductory remarks ...
The following structure should be suitable for most practical projects.

3.1 Specification

3.2 Prototype

3.3 Documentation

3.4 Critical Appraisal

...

4 Conclusions

(approx 400 words)
In the conclusion, I want a critical reflection on the content of the course. Step back from the technical details. How does the course fit into the wider world of programming languages and software engineering?

References

(g) [PL] Programming Languages 2022, Chapman University, 2022.