PROBLEM SOLVING AND PROGRAMMING I - 2020

Individual Assignment 1

CHO, Yoon

1909355@buckingham.ac.uk

Question 1.

Input **FileSize** in characters

Process **FileSize** / 960 = Time in second

Output **Time** (time taken to send the file)

Question 2.

Store **UserPIN** in system

Input **PIN**

Process 1

IF **UserPIN** == **PIN**

If !=, two more chances

If !=, !=, HALT

If ==, proceed to Process 2

Process 2 – While true until **E**

Input ;

If **O** as open an account

Input : **Initial\_Deposit** as an amount

If **Initial\_Deposit** =< **Min\_Deposit**

Output : Prints **Initial\_Deposit**

If **B** as a balance inquiry

Output: Prints the **Balance**

If **D** as a deposit

Input : **D\_Amount** as an amount

Output : Prints **Old\_Balance** and **Balance** = **Old\_Balance** + **D\_Amount**

If W as a withdrawal

Input : **W\_Amount** as an amount

If **W\_Amount** =< **Balance**

Output : Prints **Balance** = **Balance** – **W\_Amount**

If **I** as an interest

Input : **Interest\_Rate** as the rate and **Year** (use loop) as years of balancing

Output : Prints **Balance** = **Balance** \* (**Interest\_Rate** + 1) ^ **Year**

If **E** as an exit

Output : HALT

If else, ERROR MESSAGE

Question 3.

Input **In\_Rows** as number of rows and **Char** as a character to use

As C++ start counting from 0, **Rows** = **In\_Rows -1**

Process 1

Row numbers = **i**

= from 0 to **Rows,** incrementing

Output “/n” (hence change line)

Blank numbers = **j**

= each line from 0 to **Rows** – **i**

After each line, **j** increments

Output “ ”

Char numbers = **k**

= each line from 0 to **i**

After each line, **k** increments

Output **Char**

Process 2

Row numbers = **i**

= from 0 to **Rows,** incrementing

Output “/n” (hence change line)

Blank numbers = **j**

= each line from 0 to **i**

After each line, **j** increments

Output “ ”

Char numbers = **k**

= each line from 0 to **Rows - i**

After each line, **k** increments

Output **Char**

Process 3

Row numbers = **i**

= from 0 to **Rows,** incrementing

Output “/n” (hence change line)

Blank numbers = **j**

= each line from **Rows – 1** to 0 ; j=i whilst j<= **Rows -1**, hence j can increment

After each line, **j** increments

Output “ ”

Char numbers = **k**

= each line from 0 to **Rows**

After each line, **k** increments

Output **Char**

Question 4.

-a)

Input **n**

Process

(-1)^(n-1) = pow((-1),**n**-1) --- **x**

**n**! = for int **k** from 1 to **n** whilst incrementing, \*=**k** --- **y**

Hence **x/y**

Output

Sum(**x/y**) for int **n** from 1 to **n** whilst incrementing

-b)

Input **n**

Process

**n** = **Input** as n is going to be altered in order to find the last digit

If n != 0 ; recursive function

**x** = **n** % 10 hence this will continuously be the last digit

**sum** += pow(**x**,3)

**n** /= 10 so the input ditches the last digit to calculate the next left digit

Output

If **sum** == **Input**

“Armstrong number”

If **sum** != **Input**

“Not Armstrong number”

-c)

Input **a, b** in decimal

Process 1 (to binary) ; recursive function

If **a, b** != 0

**a** (or **b**) % 2

store it in array ie **a[i]** (or **b[i]**) where i starts from 1 incrementing --- LOOP

If **a** (or **b**) / 2 = 0

HALT

Else LOOP

Hence

Binary of **a** = **binom\_a = a[i]** for i from 0 to i-1, **b** = **binom\_b = b[i]** for i from 0 to i-1

Process 2 (xnor operation) ; recursive function

If **binom\_a ==** 0 && **binom\_b** == -

Return 1

**x** = **binom\_a** % 10 and **y** = **binom\_b** % 10 hence this will continuously be the last digit

if **x == y**

store 1 to array, **k[i]**

if **x** != **y**

store 0 to array, **k[i]**

**binom\_a** /= 2 and **binom\_b** /= 2 so the input ditches the last digit to calculate the next left digit

Print **k[i]** for i from 0 to i-1

Process 3 (back to decimal)

If **k[i]** != 0 whilst **i** = 0

rem = **k[i]** % 10

**k[i]** /= 10

**Deci\_XNOR** += rem\*pow(2,i)

Output **Deci\_XNOR**

-d)

Input **n**

Process ; recursive function

**x** = **n** % 10 hence this will continuously be the last digit

store **x** to **k[i]**

**n** /= 10 so the input ditches the last digit to calculate the next left digit

Output

**k[i]** for i from i to 0 hence reverse the digits