**Please, upload your answers in Canvas: assignment PC LAB – March 1**

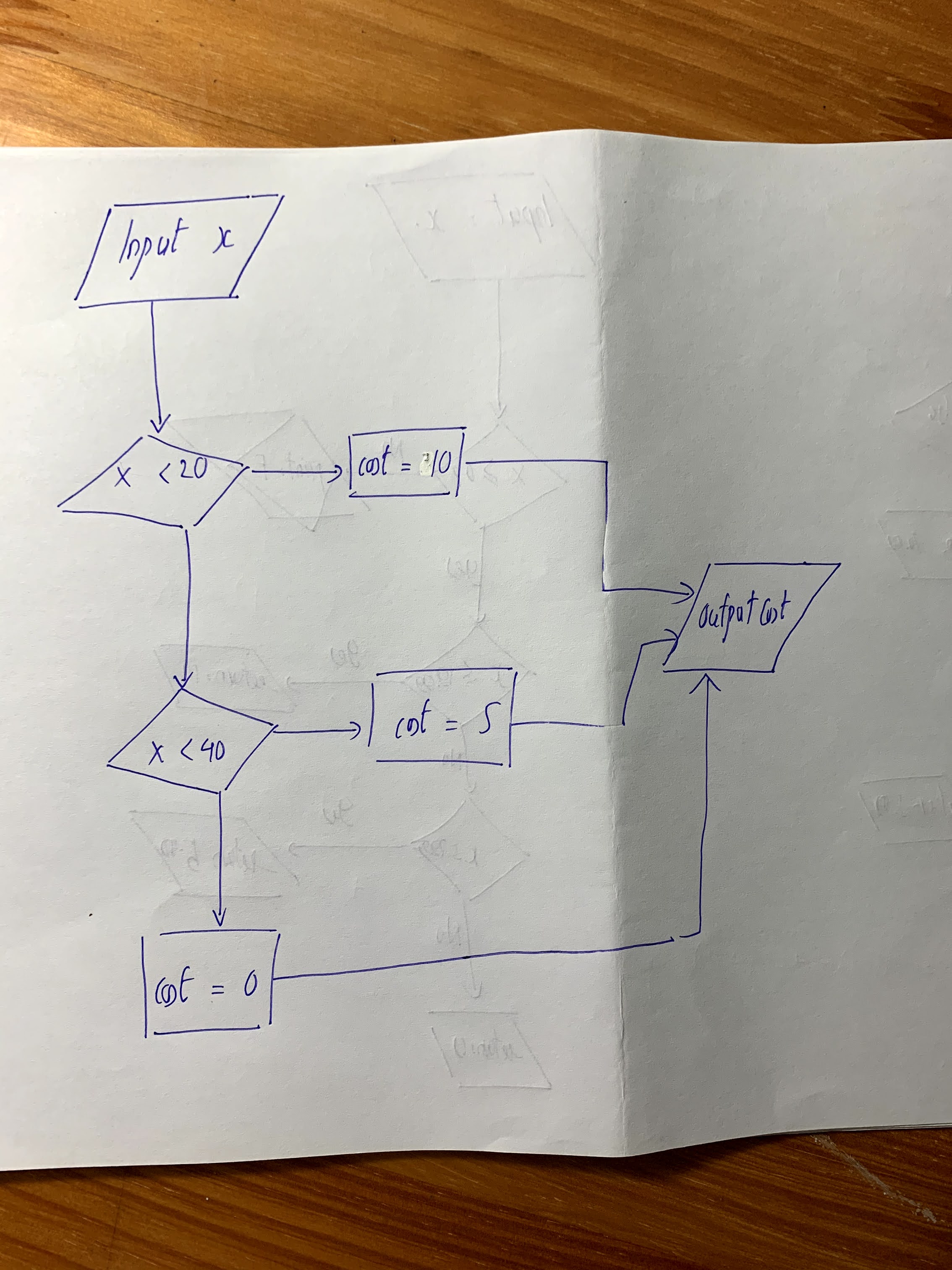
Name: Vu The Doan

Student nr: 12918687

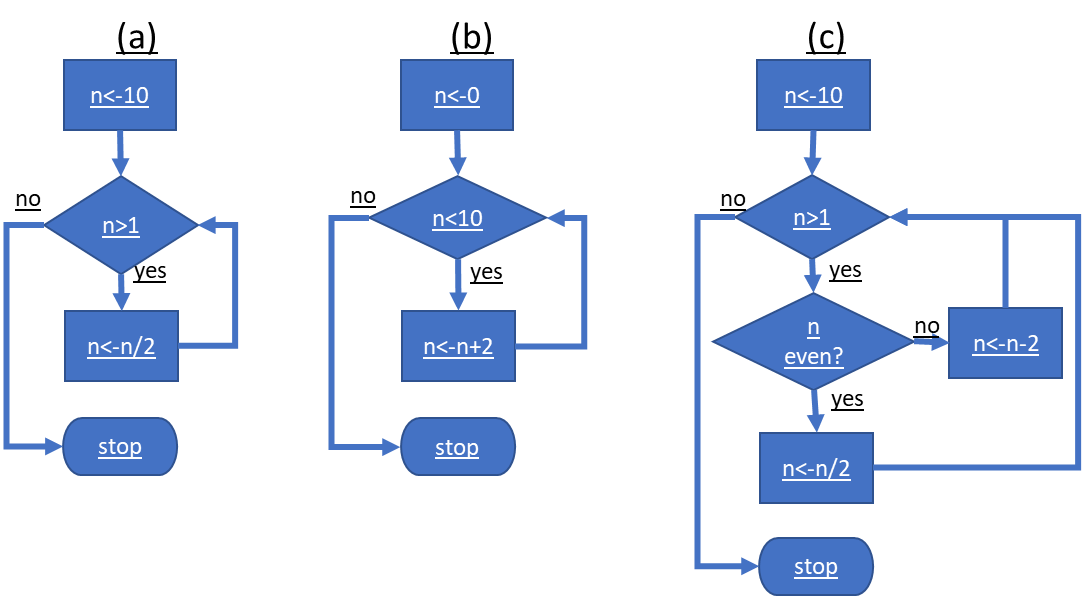
**PC1b: Flowcharts and Matrices**

**1**. Write a flowchart that describes that determines the shipping cost given the order amount given the following table (you can create a flowchart using PowerPoint Insert->Shapes->Flowchart):

|  |  |
| --- | --- |
| Shipping Cost Policy | |
| Order Amount | Shipping Cost |
| €0.00 up to €19.99 | €10.00 |
| €20.00 up to €39.99 | €5.00 |
| €40.00 up to | €0.00 |



**2**. For each of the following flowcharts, list each action that the computer takes and indicate the state of the memory after each statement in a table.



1. Table:

|  |  |  |
| --- | --- | --- |
| **Step** | **Action** | **State of memory** |
| n <- 10 | Assign value 10 to n | n = 10 |
| While loop: divide n in half until n < 1 | While loop: divide n in half until n < 1 | n = 5 |
|  |  | n = 2.5 |
|  |  | n = 1.25 |
|  |  | n = 0.625 |

1. Table:

|  |  |  |
| --- | --- | --- |
| **Step** | **Action** | **State of memory** |
| n <- 0 | Assign value 0 to n | n = 0 |
| While loop: add 2 to n until n > 10 | While loop: add 2 to n until n > 10 | n = 2 |
|  |  | n = 4 |
|  |  | n = 6 |
|  |  | n = 8 |
|  |  | n = 10 |
|  |  | n = 12 |

1. Table

|  |  |  |
| --- | --- | --- |
| **Step** | **Action** | **State of memory** |
| n <- 10 | Assign value 10 to n | n = 10 |
|  |  | n = 5 |
|  |  | n = 3 |
|  |  | n = 1 |

**3.** The following iterative sequence is defined for the set of positive integers:

n → n/2 (n is even)

n → 3n + 1 (n is odd)

Using the rule above and starting with 13, we generate the following sequence:

13 → 40 → 20 → 10 → 5 → 16 → 8 → 4 → 2 → 1

It can be seen that this sequence (starting at 13 and finishing at 1) contains 10 terms. Although it has not been proved yet (Collatz Problem), it is thought that all starting numbers finish at 1.

Which starting number, under one million, produces the longest chain?

**Answer**: My codes runs fine for m under 10^4, but took too long for 10^5 and onwards.

NOTE: Once the chain starts the terms are allowed to go above one million.

Due to overlapping sequences, we don’t have to calculate all Collatz sequences.

Let the vector len[x] denotes the length when starting at x. If len[x] already has been determined, then we know the remainder length when we arrive at x!

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | 1 | 2 | 3 | 4 | 5 |
| len[x] | 1 | 2 | 8 | 3 | 6 |

If we want to know the length of the Collatz sequence starting at 7, we get

7 → 22 → 11 → 34 → 17 → 52 → 26 → 13 → 40 → 20 → 10 → 5. Hence, we know that len[7]<-11+len[5]=11+6=17!

Hints:

- define m=1000000 as a million

- declare a 1xm vector len and fill the first elements with the table above

- use a for loop for n from 6 up to including m

- for each value of n use a while loop that to determine the Collatz sequence until a known value of len[x] is found

- calculate len[n] before going to the next value

- determine the maximum value of len[n] using a for loop

**4**. The Game of Life: In Conway’s game of life, some cells of a grid are dead (=0) and some are alive (=1). Two cells neighbour each other if they are adjacent horizontally, vertically or diagonally. At the end of each day:

– a dead cell becomes alive if it had exactly 3 living neighbours during the day

– a living cell becomes dead unless it had exactly 2 or 3 living neighbours during the day.

In the following grid, a 1 indicates a living cell.

> x=matrix(c(1,0,0,0,1,1,0,1,0,0,0,0,0,1,1,0,1,0,1,0,0,0,1,0,1),5,5)

> x

[,1] [,2] [,3] [,4] [,5]

[1,] 1 1 0 0 0

[2,] 0 0 0 1 0

[3,] 0 1 0 0 1

[4,] 0 0 1 1 0

[5,] 1 0 1 0 1

Write a program to determine how many living cells will there be the next day?

Hint: Try to determine the number of living cells first by hand.

You can use the incomplete script below. The main idea is to check the number of living neigbours by calling the function Neighbours. To determine the number of neighbours, you have to check the boundary conditions. In the main loop, call the function Neighbours for every cell. Let y denote the situation next day. Depending on whether the cell is dead or alive, determine the value of y[i,j].

Neighbours<-function(x,i,j){

living<-0

if( (i-1)>0 & (j-1)>0 ) living<-living+x[i-1,j-1]

…

if( (i+1)<6 & (j+1)<6 ) living<-living+x[i+1,j+1]

return(living)

}

x=matrix(c(1,0,0,0,1,1,0,1,0,0,0,0,0,1,1,0,1,0,1,0,0,0,1,0,1),5,5)

y <- matrix(0,5,5)

for(i in 1:5){

for(j in 1:5){

….

}

}

y

sum(y)

How many living cells will there be the next day?

**Answer**: 9

R-code:

getwd()

setwd("D:/UvA/Year 1/Block 5/Programming and Numerical Analysis/Week 1")

rm(list = ls())

m <- 10^5

a <- c(1,2,8,3,6)

len <- c(rep(0, times = m))

len[1:5] <- a

for (n in 6:m) {

x <- n

count <- 0

while (x != a & x >=n){

if (x %% 2 == 0) {

x <- x/2

count <- count + 1

}

else {

x <- (x\*3)+1

count <- count + 2

}

count <- count + len[x]

len[n] <- count

}

}

b <- which.max(len)

print(b)

#4

Neighbours<- function(x,i,j){

living<-0

if( (i-1)>0 & (j-1)>0) living<- living + x[i-1,j-1]

if( (i-1)>0 & (j)>0) living <- living + x[i-1,j]

if( (i-1)>0 & (j+1)<6) living <- living + x[i-1,j+1]

if( (i)>0 & (j-1)>0) living <- living + x[i,j-1]

if( (i+1)<6 & (j+1)<6) living <- living + x[i,j+1]

if( (i+1)<6 & (j-1)<0) living <- living + x[i+1,j-1]

if( (i+1)<6 & (j)>0) living <- living + x[i+1,j]

if( (i+1)<6 & (j+1)<6) living <- living+ x[i+1,j+1]

return(living)

}

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

y <- matrix(0,5,5)

for(i in 1:5){

for(j in 1:5){

if(x[i,j]==0){

if(Neighbours(x,i,j)==3) y[i,j]<-1

}

if(x[i,j]==1){

count<- Neighbours(x,i,j)

if((count==2)|(count==3)) y[i,j]<-1

}

}

}

y

sum(y)