Name:

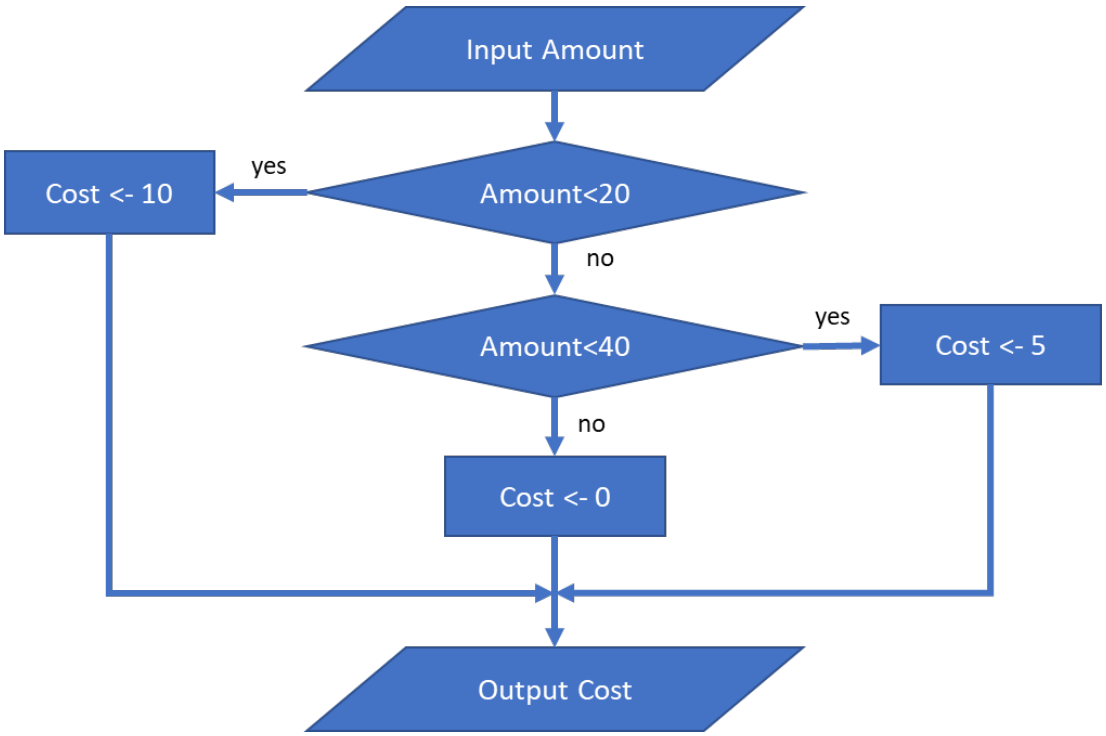
Student nr:

PC1b: Flowcharts and Matrices

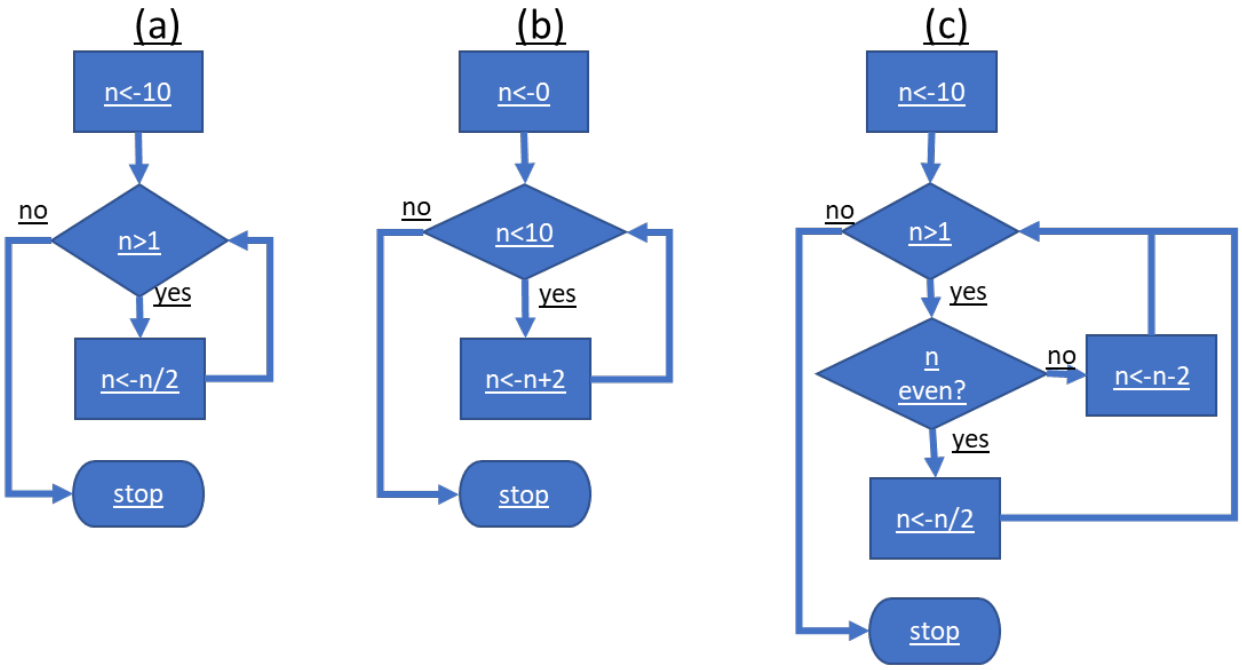
1. Write a flowchart that describes that determines the shipping cost given the order amount given the following table (you can create a flowchart using Powerpoint Insert->Shapes->Flowchart):

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

Below, one possible Flowchart. Other flowcharts could also be correct!



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.



a. `n <- 10, n <- 5, n <-2.5, n <- 1.25, n <- 0.625`
b. `n <- 0, n<- 2, n <- 4, n <-6, n <-8, n <-10`
c. `n <- 10, n <- 5, n <-3, n<-1`

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

$n \rightarrow n/2$ (n is even)
$n \rightarrow 3n + 1$ (n is odd)

Using the rule above and starting with 13, we generate the following sequence:
$13 \rightarrow 40 \rightarrow 20 \rightarrow 10 \rightarrow 5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$

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: 837799

```
# initialize
m <- 1000000L
len <- c(1,2,8,3,6, rep(0L, m-5))

# calculate the rest of vector len
for(n in 6:m){

  # initialize
  x <- n
  count  <- 0

  # calculate len[n]
  while (x >= n) {

    # initialize
    count <- count + 1

    # calculate the Collatz successor of x
    if(x %% 2 == 0) {
      x<- x %/% 2
    } else {
      x <- 3*x + 1
    }

    # test for a cycle
    if (x == n) {
      stop(paste("Starting value",n,"leads to a cycle!",sep="
"),       call.=FALSE)
    }

  }

  # populate len[x]
  len[n] <- count+len[x]

}

# finalize: find and dump results

  # find the starting value with the highest number of steps
    # initialize
    steps <- 0
    n_start <- 0

    # update steps and n_start
    for(n in 1:m){
      if(len[n]>steps){
        steps <-len[n]
        n_start <- n
      }
    }

  # print result
  cat("Starting value ",n_start," takes ",steps," steps.")
```

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 live cells will there be the next day?

How many live cells will there be the next day?
Answer: 8

```
#Game of Life
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)

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

Output:

```
> y
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    0    0    0    0
[2,]    1    1    1    0    0
[3,]    0    0    0    0    1
[4,]    0    0    1    0    1
[5,]    0    1    1    0    0
> sum(y)
[1] 8
```