Name: Shahriar Ahmed.
ID: 20101588.
Section: 04.
Topic: Theory Assignment - 2.

Task-1:

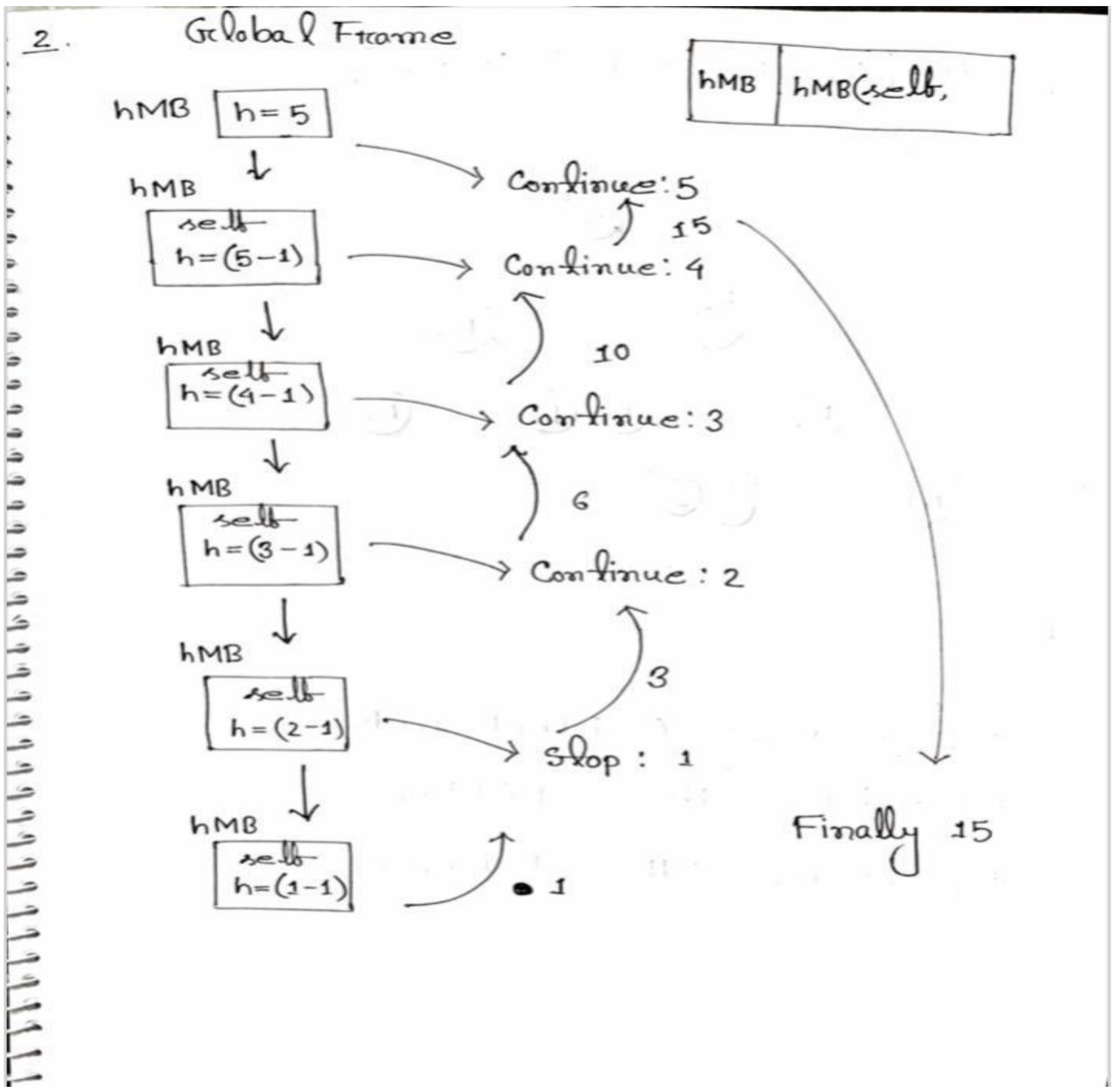
```
def powerN(base, power):
    if power == 0:
        return 1
    elif power == 1:
        return base
    else:
        return (base * powerN(base, (power-1)))

base = int(input("Enter the base number: "))
power = int(input("Enter the power number: "))
value = powerN(base, power)
print("The Value: ", value)
```

Task-2:

Code:

```
class Trace:
    def hMB(self,h):
        if (h==0):
            print("Stop: ",h)
            return 0
        elif(h==1):
            print("Stop: ",h)
            return h
        else:
            print("Continue: ",h)
            return h + self.hMB(h-1)
#Tester
t = Trace()
print("Finally ", t.hMB(5))
```

Recursive Flow Diagram:



Task-3:

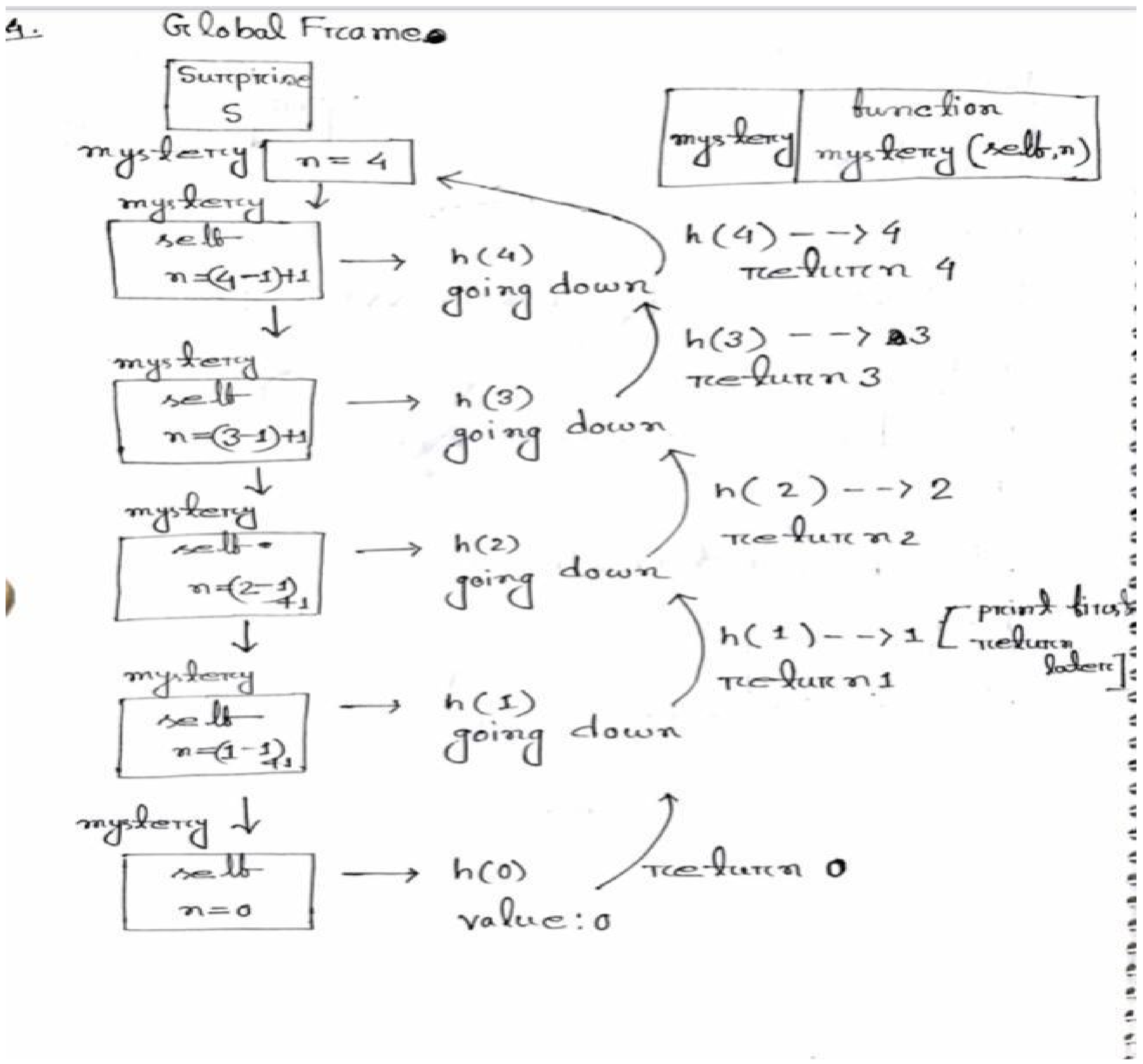
```
def hocBuilder(height):
    if height == 0:
        return 0
    elif height == 1:
        return 8
    else:
        return 5 + hocBuilder(height-1)

print(hocBuilder(3))
```

Task-4:

Code:

```
class Surprise:
    def mystery(self,n):
        print("h(" ,n,")")
        if(n==0):
            print("value: 0")
            return 0
        else:
            print("going down")
            temp = self.mystery(n-1)+1
            print("h(",n,") --> ",temp)
            return temp
#Tester
s = Surprise()
s.mystery(4)
```

Recursive Flow Diagram:



Task-5:

a.

```
def col(i,j):
    if(j<=i):
        print(j+1,end=" ")
        col(i,j+1)
def sol(n,i,j):
    if(i<n):
        col(i,j)
        print()
        sol(n,i+1,0)
```

```
n=int(input())
sol(n,0,0)
```

b.

```
def col(i,j):
    if i+1==n and j<=i:
        print(n-j,end=" ")
        col(i,j+1)
    else:
        if(j<=n-i-2):
            print(" ",end=" ")
            col(i, j + 1)
        elif(j>n-i-2 and j<n):
            print(n-j,end=" ")
            col(i, j + 1)

def sol(n,i,j):
    if(i<n):
        col(i,j)
        print()
        sol(n,i+1,0)

n=int(input())
sol(n,0,0)
```

Task-6:

```
class FinalQ:
    def print(self,array,idx):
        if(idx<len(array)):
            profit = self.calcProfit(array[idx])
            print("{}. Investment : {}; Profit : {}".format(idx+1, array[idx], profit))
            return self.print(array,idx+1)

    def mult(self, n, m):
        if m <= 0:
            return 0
        elif m<1:
            return n/2 + self.mult(n, m-1)
        else:
            return n + self.mult(n, m-1)

    def calcProfit(self,investment):
        if investment <= 25000:
            return float(0)
        elif investment > 25000 and investment <= 100000:
            return float(self.mult(75000, 4.5))/100
        else:
            extra = investment - 100000
            return float(self.mult(75000, 4.5)) /100 + float(self.mult(extra, 8)) / 100

#Tester
array=[25000,100000,250000,350000]
f = FinalQ()
f.print(array,0)
```

Task-7:

```
def binarySearch(arr, start, end, x):
    while start <= end:
        mid = start + (end - start) // 2
        if arr[mid] == x:
            return len(arr[0:mid+1])
        elif arr[mid] < x:
            start = mid + 1
        else:
            end = mid - 1
            if arr[mid]> x and arr[mid-1]<=x:
                return len(arr[0:mid])

def Q7(arr,brr):
    for x in brr:
        print(binarySearch(arr,0,len(arr),x),end=" ")


a,b=map(int,input().split())
arr = list(map(int,input().split()))
brr=list(map(int,input().split()))
Q7(arr,brr)
```

Task-8:

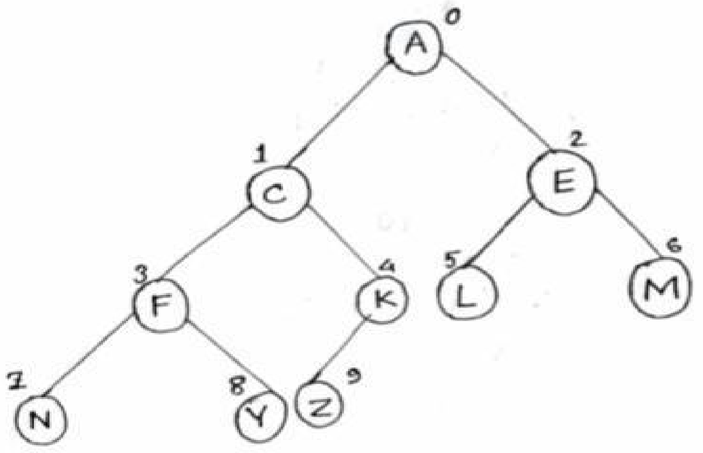
```
def friday_fun(players, string, i, j, counter, checker):
    if counter+1 < players:
        if string[j]=="1" or string[j]=="6" or string[j]=="3" or string[j]=="5":
            j+=1
            j=j%players
        else:
            checker[i%players]=True
            counter+=1
        friday_fun(players, string, i+1, j, counter, checker)
    else:
        print(checker.index(0)+1)

i, j, counter = 0, 0, 0
players = int(input())
checker = [False] * players
string = input()
string =list(string)
friday_fun(players, string, i, j, counter, checker)
```

Task-9:

9.

A = [A, C, E, F, K, L, M, N, Y, Z]

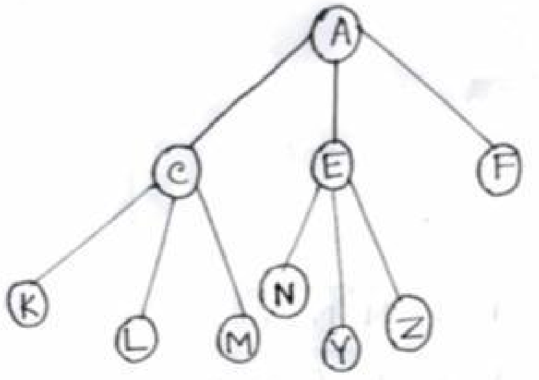
a.



b.

* preorder ⟶ ACFNYKZELM
* inorder ⟶ NFYCZKALEM
* postorder ⟶ NYFZKCLMEA

c. A = [A, C, E, F, K, L, M, N, Y, Z]



d.

* pre order ⟶ ACKLMENYZF

* in order ⟶ KCLMANEYZF

* post order ⟶ KLMCNYZEFA

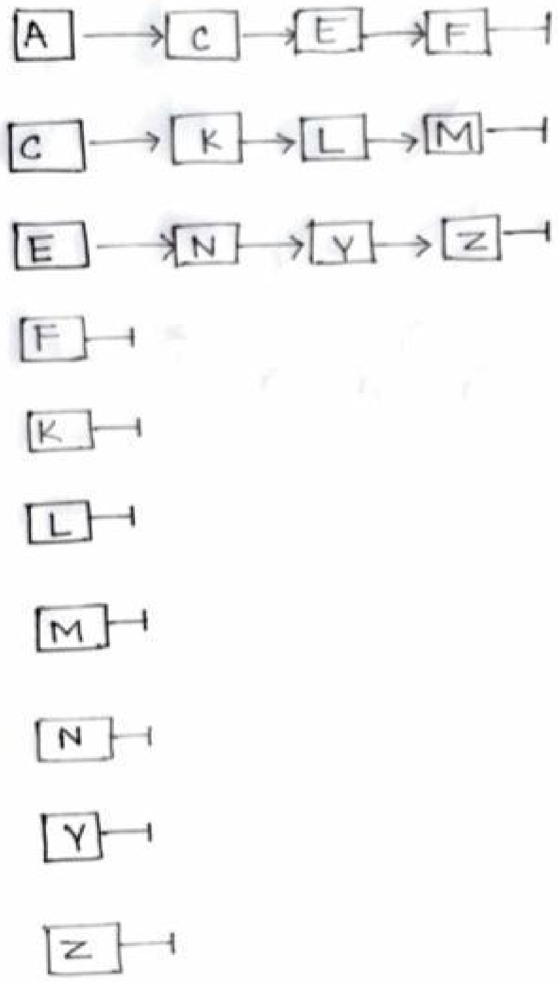
e.

* Adjacency Matrix:

	A	C	E	F	K	L	M	N	Y	Z
A	0	1	1	1	0	0	0	0	0	0
C	0	0	0	0	1	1	1	0	0	0
E	0	0	0	0	0	0	0	1	1	1
F	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0
Y	0	0	0	0	0	0	0	0	0	0
Z	0	0	0	0	0	0	0	0	0	0

* Adjacency List:

outgoing:


A C E F
C K L M
E N Y Z
F
K
L
M
N
Y
Z