Krity Haque Charu

19101173

Lab Section: 05

Assignment 01

```
graph_matrix=[]
with open ('/content/input.txt') as f:
  lines=f.readlines()
  #print(lines)
for line in lines:
  v=line.split()
  #print(v) #Each line became a list
  graph_matrix.append(v)
#print(graph_matrix)

Y_list=[]
visited=[]
hightest_affected=[]

row=len(graph_matrix)
#print(row)
col=len(graph_matrix[0])
#print(col)

for i in range (row):
  for j in range (col):
    #print(graph_matrix[i][j])
    if (graph_matrix[i][j]=="Y"):
      graph_matrix[i][j]=1
      Y_list.append((i,j))
    else:
      graph_matrix[i][j]=0

#print(graph_matrix)
#print(Y_list)
s=[]
for i in range(row):
  for j in range(col):
    if (graph_matrix[i][j]==1 and graph_matrix[i][j] not in visited):
      s.append(dfs(graph_matrix,i,j))
print(max(s))
```

```
def dfs(graph,r,c):
  if r<0 or c<0 or r>=row or c>=col:
    return 0
  if(graph[r][c]==0):
    return 0
  graph[r][c]=0
  size=1
  for i in range(r-1,r+2,1):
    for j in range(c-1,c+2,1):
      if(i != r or j != c):
        size+=dfs(graph,i,j)
  return size
```

Assignment 02

```
g_matrix=[]
with open ("/content/Question2 input1.txt") as file :
    lines = file.readlines()
rownum=lines[0]
colnum=lines[1]
lines=lines[2:]
for line in lines :
    v=line.split()
    g_matrix.append(v)


alli_list=[]
hum_list=[]
trap_list=[[]]
times=[]

row=len(g_matrix)
col=len(g_matrix[0])


for i in range (row):
    for j in range (col) :
        if (g_matrix[i][j])=='A' :
            g_matrix[i][j] = 0
            alli_list.append((i,j))
```

```
        elif (g_matrix[i][j])=='H' :
            g_matrix[i][j] = 1
            hum_list.append((i,j))
        else :
            g_matrix[i][j] = 2


for alien in alli_list:
  times.append(bfs(g_matrix,alien))
print("time = "+ str(max(times)))
count=0
for i in range (row):
  for j in range(col):
    if(g_matrix[i][j]==1):
      count=count+1
if(count==0):
  print("There is no human left")
else:
  print(str(count)+ " Human Alive")



def bfs(graph,start):
  queue=[]
  i=start[0]
  j=start[1]
  queue.append(start)
  #print(start)
  graph[i][j]=2
  time=0
  while len(queue)>0:
    a,b=queue.pop(0)
    #print(a,b)
    #print(graph[a][b])
    Yes=0
    for m,n in [[1,0],[0,1],[-1,0],[0,-1]]:
      if boundaryok(graph,a+m,b+n):
        queue.append((a+m,b+n))
        #print(queue)
        graph [a+m][b+n]=2
        Yes=1
    if Yes==1:
      time=time+1
```

```
    #print(graph)
  return(time)


def boundaryok(graph,a,b):
  if a<0 or b<0 or a>=row or b>=col:
    return 0
  if(graph[a][b]==2):
    #print("hello")
    return 0
  return 1
```