Practical 3(4)

1 Question 1

Would you use a stack or a queue for the following applications?

(a) Managing a waiting list - **Queue**

(b) Recording pages visited by a web browser for use with a back button - **Stack**

(c) Multiprogramming (A computer system used by multiple users) - **Queue**

(d) Undo sequence in a text editor - **Stack**

2 Question 2

Given an initially empty queue complete the table below with the output of each operation and the updated contents of the queue

operation	output	queue
enqueue(4)		4
enqueue(6)		6,4
dequeue	4	6
enqueue(2)		2,6
dequeue	6	2
top	SYNTAX ERROR	
dequeue	2	
dequeue	ERROR	
isEmpty	TRUE	
enqueue(1)		1
enqueue(2)		2,1
size	2	2,1
enqueue(6)	6,2,1	
enqueue(4)		4,6,2,1
dequeue	1	4,6,2

3 Question 3

Given an initially empty stack complete the table below with the output of each operation and the updated contents of the stack

operation	output	stack
push(2)		2
push(9)		9,2
push(8)		8,9,2
pop	8	9,2
pop	9	2
top	2	2
pop	2	
pop	ERROR	
isEmpty	TRUE	
push(1)		1
push(3)		3,1
push(5)		5,3,1
size	3	5,3,1
push(2)		2,5,3,1
pop	2	5,3,1

4 Question 4

An arithmetic expression written in postfix notation is evaluated by reading from left to right and when an operator (plus, minus, multiply, divide) is met it is evaluated on the last two operands (numbers) seen, and the value obtained is substituted in to the expression. For example: to evaluate

$$4512+-/$$

we read across until we reach the + which we then evaluate on the last two numbers: 1 and 2. The result obtained is 3 which replaces them in the string to give

$$453-/$$

We now read the minus sign and evaluate it on the 5 and 3 (keeping that ordering) to get

$$42/$$

and so finally we evaluate 4/2 to obtain the result 2. Note that in the usual infix notation we would have written this as

$$4/(5-(1+2))$$

Write pseudocode that evaluates a postfix expression. Use a stack.

```
stack S
def push(e)
    # If size = N then the stack is full
    if size = N then
        throw a FullStackException
    end if
    t=t+1
    S[t]=e

def pop()
    if isEmpty then
        throw a EmptyStackException
    end if
    e = S[t]
    S[t] = NULL
    t = t - 1
    return e

def top()
    if isEmpty then
        throw a EmptyStackException
    end if
    return S[i]
for i in string:
    if i is an operator then
        var1=pop()
        var2=pop()
        push (var2 operator var 1)
    else
        push(i)
print(top(S))
```

5 Question 5

```
#Stack to put all the information on
stack S
#Queue to store the values after they have been removed from the stack
```

```
queue Q
#A list that has all the prices on
list L
def push(e)
    # If size = N then the stack is full
    if size = N then
        throw a FullStackException
    end if
    t=t+1
    S[t]=e

def pop()
    if isEmpty then
        throw a EmptyStackException
    end if
    e = S[t]
    S[t] = NULL
    t = t - 1
    return e

def top()
    if isEmpty then
        throw a EmptyStackException
    end if
    return S[i]

def enqueue(e)
    if size=N-1 then
        throw a FullQueueException
    end if
    Q[r]=e
    r=r+1 mod N

def dequeue()
    if isEmpty then
    throw a EmptyQueueException
    end if
    temp=Q[f]
    Q[f]=NULL
    f=f+1 mod N
    return temp

def size()
    return (N-f+r) mod N

for l in list
    for s in stack
        if Stack.isEmpty=True then
            return *
        compare=pop()
        enqueue(compare)
        if compare>i then
            return size(Q)
    for q in queue
        transfer=dequeue()
        S.push(transfer)
```

6 Question 6

```
stack S
string postfix
string infix
queue Q
def push(e)
    # If size = N then the stack is full
    if size = N then
        throw a FullStackException
    end if
    t=t+1
    S[t]=e

def pop()
    if isEmpty then
        throw a EmptyStackException
    end if
    e = S[t]
    S[t] = NULL
    t = t - 1
    return e

def top()
    if isEmpty then
        throw a EmptyStackException
    end if
    return S[i]
def enqueue(e)
    if size=N-1 then
        throw a FullQueueException
    end if
    Q[r]=e
    r=r+1 mod N

def dequeue()
    if isEmpty then
    throw a EmptyQueueException
    end if
    temp=Q[f]
    Q[f]=NULL
    f=f+1 mod N
    return temp

def size()
    return (N-f+r) mod N


List High = [*,/]
List Low = [-,+]
for i in infix
    if i=int then
        postfix=postfix+i
    else

        for s in Stack
            compare=S.pop()
```

```
            enqueue(compare)
            if i in High and compare in High:
                    postfix=postfix+compare
            elif i in Low
                postfix=postfix+compare
        for q in Q
            transfer = dequeue()
            S.push(transfer)
        S.push(i)
print(postfix)
```