Day: M T W T F S Date: __/__/__

Q.1 Write the definition of the following functions for stack, while re-utilizing the implemented functions of linked list.

⇒ Using functions of linked list.

- DECLARE (S)

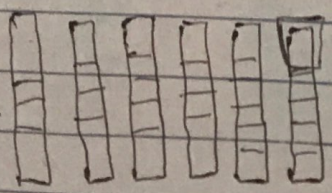
- Empty (S)

- Top (S)
  ⇒ Retrieve (First (S), S)
- Push (e, S)
  ⇒ insert (e, First (S), S)
- Pop (S)
  Retrieve (First (S), S), Delete (First (S), S

STACK:

(special type of list)

stack is a linear data structure which follows a particular order in which the operations are perform The order may be LIFO (Last come first out) or Fito (First come Last out)



Declare (S):

Declare : → S:

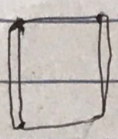
CHECKED BY: ☐ EXCELLENT ☐ GOOD ☐ NEED IMPROVEMENT BABAR PAPER PRODUCTS

Day: M T W T F S Date: __/__/____

The function value of Declare(S) is an empty stack.

Empty : → S:

The function Empty(S) causes the stack to be emptied and returns the position which is End(S)



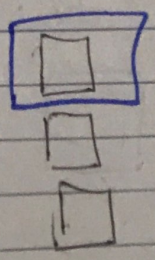
Along with Empty then is another function is Emply(S) which is a boolean function which returns true is the stack is emply, else returns false.

Top (s):

Top : S → E:

This function is defined to return the first element in the list but if the list is empty, then its value is undefined.

Top (S)

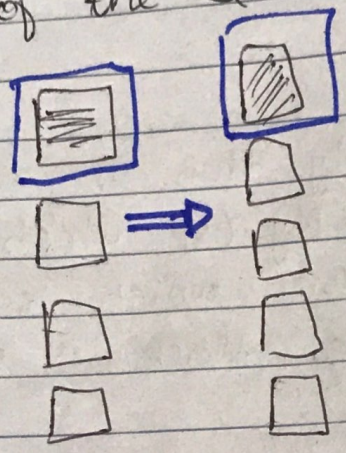


Day: M T W T F S Date: __/__/__

Push (e, s):

Push : E X S → L :
Push (e, s)

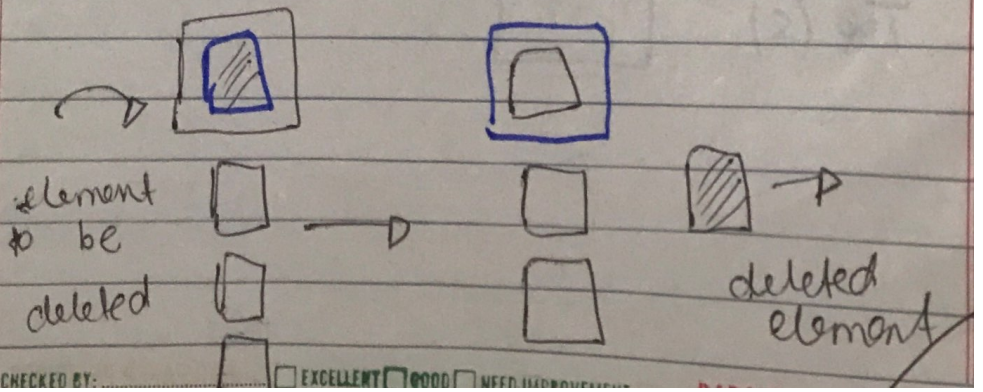
This function takes two arguments e and s element and stack which insert element (e) at the top of the stack.



Pop (s):

Pop : S → E
Pop (s) :

This function removes the top element from the stack i.e return the top element and delete it from the stack.



CHECKED BY: ..... ☐ EXCELLENT ☐ GOOD ☐ NEED IMPROVEMENT

Day: M T W T F S

Date: __/__/____

Implementation of stack in cpp using linked lists:

```
#include <iostream>
using namespace std;
type of struct Node
{   int data;
    Node * next;
} Node;
class stack
{
  Private:
   Node * top;
   int max-size;
   int size = 0;
Public:
   voide Delare (int size)
{     Max-size = size;
      top = Null;
    cout << " stack created .\n ";
}
  if (size >= Max-size)
  { cout << " Stack is full " << endl; }
else
    Node * temp;
   temp = New Node();
   temp-> data = data;
   temp -> next = top;
    top = temp;
    Size ++;
 cout << " Element pushed into the stack.\n ";
  }}
```

CHECKED BY: ..... ☐ EXCELLENT ☐ GOOD ☐ NEED IMPROVEMENT

Day: M T W T F S Date: __/__/__

```
void pop()
{
  if (size == 0)
  { cout << "stack is empty" << endl; }
else
{  Node * temp;
   temp = top;
   top = temp -> next;
   temp -> next = NULL;
   delete temp;
   size--;
cout << "Element removed from stack \n";
}
}
void Top()
{
  if (size == 0)
{ cout << "stack is empty \n"; }
else
{
 cout << Top element is = " top -> data << end
}
void size()
{
cout << "The size of the stack is " << size << endl
}
void Empty()
{   Node *temp;
  temp = top;
  top = NULL;
  size = 0
  cout << "stack emplied \n"; }
```

```
oid display()
{
  if (size == 0)
{
  cout << " Stack i
}
else
{
  Node * temp;
  temp = top;
  while (temp !
{
   cout << tem
   temp = te
}
}
};
   int
{
  int
  stac
  while
{
  cout
  cout
  co
  do
  e
```

CHECKED BY: ..... ☐ EXCELLENT ☐ GOOD ☐ NEED IMPROVEMENT BABAR PAPER PRODUCTS

Date: __/__/____

```
void display ()
{
   if (size == 0)
{
   cout << " Stack is empty \n";
}
else
{
   Node * temp;
   temp = top;
   while (temp != Null)
{
   cout << temp -> data << " , ";
    temp = temp -> Next;
}
}
};
    int main ()
{
  int choice item , s;
    Stack S;
  while (1)
{
  cout << " Stack operations " << endl;
  cout << " 1. Declare Stack " << endl;
  cout << " 2. push element " << endl;
  cout << " 3. Pop element " << endl;
  cout << " 4. Display Stack " << endl;
  cout << " 5. Empty the stack " << endl;
  cout << " 6. Size of stack " << endl;
  cout << " 7. Top item " << endl;
```

Day: M T W T F S Date: /

```
cout << " 8. Quit \n 9. Enter your ch
cin >> choice;
switch (choice)
{  case 1;
   cout << "Enter size of stack";
    cin >> s;
    s. Declare (s);
    break;
 Case 2;
cout << " Enter value to be pushed ";
cin >> item;
 s. push (item);
  break;
 Case 3:
   s. pop ();
   break;
 Case 4:
  s. Display ();
   break;
 case 5:
   s. Empty ();
   break;
 Case 6:
  s. Size ();
  break;
Case 7 : s. top (); break;
Case 8: return 0; break;
default;
 cout << " Wrong Choice " << endl;
}}
   return 0;
}
```

CKED BY: ..... ☐ EXCELLENT ☐ GOOD ☐ NEED IMPROVE

Day: M T W T F S Date: / /

UNIVERSITY OF ENGINEERING & TECHNOLOGY PESHAWAR

DEPARTMENT OF COMPUTER SYSTEM ENGINEERING.

NAme: Salman Shakeel

Section: B

Reg no: 20PwCse1925

Assignment : 02

INSTRUCTOR

SIR NASRU

DATA STRUCTURE & ALGORITHM:

Day: M T W T F S

Date: __/__

Q Write the difinition of the follo
functions for Queue while re-utili
the implemented functions of linked

Solution:-

Re-utilizing the fundions of linked lis

- Declare (Q)
- Empty (Q)
- Head (Q)
  - Retreive (First (Q), Q)
- Enqueue (e, Q)
  - Insert (e, End (Q, Q)
- Dequeue (Q)
  - Retreive (First (Q), Q)
  - Delete (First (Q), Q)

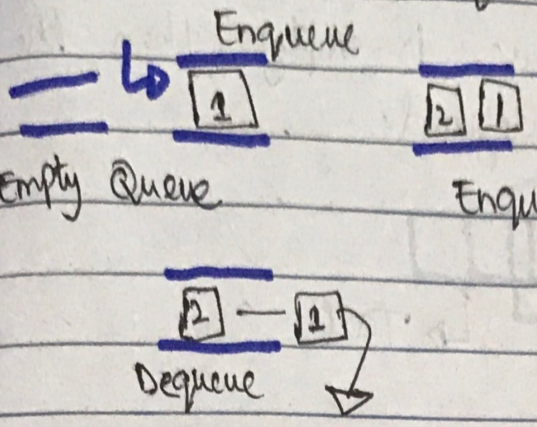
Queue:

A Queue is a useful data structure in programming. It is similar to the ticket queue outside a cinema hall, where the first person entering the queue is the first person who gets the ticket Queue follows the (FIFO) 'First in First out rule, the

Day: M T W T F S

Date: __/__/____

item that goes in first is the item that comes out first too.



In the above example since 1 was kept in the queue before 2, it was the first to be removed from the queue as well. As it follows the fifo rule.

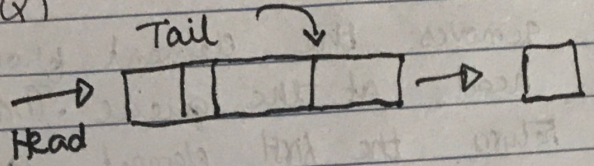
- **Declare (Q):**

  Declare: → Q:

  This function declares a new empty queue.

- **Empty (Q):**

  This function causes the queue to be emptied and it returns position End (Q).

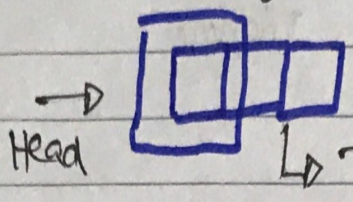


- **Head (Q):**

  Head : Q → E:

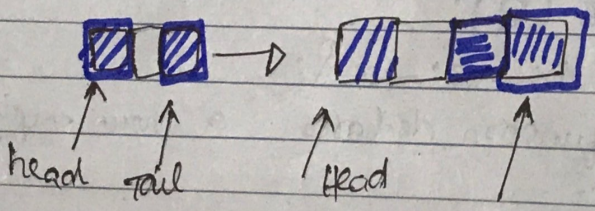
Day: M T W T F S Date: __/__/__

The function value Head(Q) is th function first element in the list. If the Queue is empty, then the value is undefined.



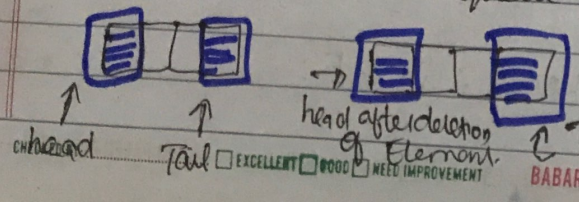
- **Enqueue (e, Q):**

Enqueue : E x Q → Q :
Enqueue (e, Q)
Add an element e in the tail of the queue.



- **Dequeue (Q):**

Dequeue : Q → E :
Dequeue (Q)
It removes the element from the head at the queue. That is it return the first element and delete it from the queue.



EXCELLENT GOOD NEED IMPROVEMENT
BABAR PAPER PRODUCTS

Day: M T W T F S Date: / /

Queue implementation in C++ code. Using linked list operations:

```
#include<iostream>
using namespace std;
typedef Node.
{
    int data;
    Node *next = Null;
} Node;
class Queue
{
  Private :
    Node * Head;
    Node* Tail;
 int Capacity, Size = 0;
Public :
    void Declare (int c)
{
   Head = tail = Null;
    Capacity = c;
    Cout << "Queue created" << endl;
}
void Enqueue (int n)
{
   if (size >= capacity)
   Cout << "Queue is full" << endl;
 else
{
    Node *temp;
   temp = new Node;
   temp => data = n;
```

Day: M T W T F S Date: __/__/

```
    if (Head == Null)
    { Head -> next = temp;  Tail = temp; }
    cout << "Element Enqueued \n";
      size++;
  }
}

Void Dequeue ()
{
  if (Front = Null)
  cout << " Queue is Empty .\n";
else
{
  Node * temp = Head;
    Head = Head -> Next;
  if (Head == Null)
  {
    Tail = Null; }
  delete temp;
  size--;
Cout << " Element Dequed \n";
}
}

void Head ()
  {
    if (Head == Null)
  cout << " Queue is empty .\n";
else
  cout << " Head " << Head -> data << endl;
}
    void size ()
    if (Head == Null)
```

CHECKED BY: ..... ☐ EXCELLENT ☐ GOOD ☐ NEED IMPROVEMENT

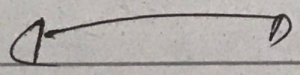
```
    cout << " Queue is Empty .\n";
else
{ Node * temp = Head;
   Node * N;
  while (temp != Null)
  {
     N = temp;
     temp = temp -> Next;
       delete N;
}
  Head = tail = Null;
  cout << " Queue Emptied .\n";
    size = 0;
}
}

   void   Display ()
  {
     if (Head == Null)
  { cout << " Queue is empty \n "; }
else
{ Node * temp = Head;
   int Array [size];
     int i = 0;
   while (temp != Null)
{
```

```
        Array [i] = temp -> data;
        i++;
        temp = temp -> Next;
    }
    for (int j = size-1 ; j > 0; j--)
    {
      cout << Array [j] << "->";
    }
      cout << Array [0] << endl;
    }
  };
    int main()
{
    int choice, item ,s;
    Queue q;
   while (1)
{
cout << "1. Create Queue" << endl;
cout << "2. Enqueue" << endl;
cout << "3. Dequeue" << endl;
cout << "4. Empty the Queue" << endl;
cout << "5. Size of Queue" << endl;
cout << "6. Head" << endl;
cout << "7. Tail": << endl;
```

```
cout << "8. Display Queue" << endl;
cout << "9. Quit" << endl;
cout << "Enter choice" << endl;
cin >> choice;
switch (choice)
{
 case 1;
cout << "Enter Size of Queue" << endl;
cin >> s;
 q. Declare (s);
 break;
case 2;
cout << "Enter the value of Enqueued.\n";
cin >> item;
q. Enqueue (item);
   break;
 case 3:
 q. dequeue ();
  break;
case 4:
 q. Empty ();
  break;
case 5;
  q. size ();
 break;
```

```
case 7:
  q.tail();
    break;
case 8:
  q.display();
    break;
case 9:
    return 0;
    break;
default:
    cout<<"wrong choice.?\n";
}
}
return 0;
}
```



End.