STACKS, QUEUES AND LINKED LIST

Stack

In computer science, a stack is a Last in, First out (LIFO) data structure. It simply means that an element that is inserted at the end will be deleted first. To Manage a stack all the insertion and deletion takes place from one position/end called "top".

One of the common uses of stack is in function call.

Operations on the Stack

There are two fundamental operations

Push

Pop

Push means to insert an element

Pop means to delete an element

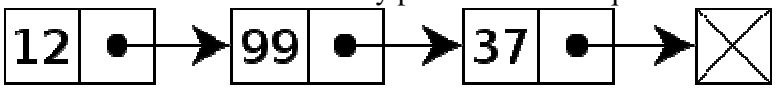
Queue

In computer science, a Queue is a First in, First out (FIFO) data structure. It simply means that an element that is inserted at the beginning will be deleted first. To manage a queue all the insertion and deletion takes place from two different positions called "front" and "rear".

Every element is inserted from the rear position and deleted from the front position in the queue.

Linked List

A linked list is a data structure consisting of a group of nodes which together represent a sequence. Under the simplest form, each node is composed of a data and a reference (in other words, a *link*) to the next node in the sequence; more complex variants add additional links. This structure allows for efficient insertion or removal of elements from any position in the sequence.



Here in the figure is an example of a linked list whose nodes contain two fields: an integer value and a link to the next node. The last node is linked to a terminator used to signify the end of the list.

Linked lists are among the simplest and most common data structures. They can be used to implement several other common abstract data types, stacks, queues etc.

The principal benefit of a linked list over an array is that the list elements can easily be inserted or removed without reallocation or reorganization of the entire structure because the data items *need not be stored contiguously* in memory or on disk. Linked lists allow insertion and removal of nodes at any point in the list, and can do so with a constant number of operations if the link previous to the link being added or removed is maintained during list traversal.

Linked list are dynamic structure where memory allocation takes place at run time.

Operation on a linked list

There are three basic operations on a linked list

Insertion

Deletion

Traversal

Inserting a node or element into Linked list :

Inserting an element into linked list contains 3 types .

1. Insertion at beginning of the Linked list
2. Insertion after/before any element of the linked list
3. Insertion at the end of the linked list

Deleting a node from the Linked list.

A node can be deleted in 3 ways similar to Insertion.

1. Deleting a Node from the beginning of the Linked List
2. Deleting a node before/after an element from the Linked list.
3. Deleting a node from the end of the Linked List .

Implementation of stacks using a linked list

The stack which is implemented using linked list is called linked stack or dynamic stack

```
#include<iostream.h>
#include<conio.h>
struct node
{
int data;
node * next;
};
class stack
{
node *top;
public:
stack()
{
top=NULL;
}
void stackpush();
void stackpop();
void displaystack();
};
```

```
void stack::stackpush()
{
node *ptr;
ptr=new node;
cout<<"Enter the element to be pushed"<<endl;
cin>>ptr->data;
if(top==NULL)
 ptr->next=NULL;
else
 ptr->next=top;
top=ptr;
}
void stack ::stackpop()
{
 node *ptr;
ptr=top;
 cout<<"The popped element is "<<ptr->data;
 top=top->next;
 delete ptr;
}
```

```
void stack :: displaystack()
{
node *ptr;
ptr=top;
cout<<"The stack is "<<endl;
while(ptr!=NULL)
{
cout<<ptr->data<<endl;
ptr=ptr->next;
}
}
```

```
void main()
{
clrscr();
char ans;
stack s1;
do
{
s1.stackpush();
cout<<"wish to continue "<<endl;
cin>>ans;
}while(ans=='y');
s1.displaystack();
cout<<"Press any key to pop an
element"<<endl;
getch();
s1.stackpop();
s1.displaystack();
getch();
}
```

Implementation of queues using a linked list

The queue which is implemented using linked list is called linked queue or dynamic queue

```
#include<iostream.h>
#include<conio.h>
struct node
{
int data;
node * next;
};
class queue
{
node *front,*rear;
public:
queue()
{
rear=front=NULL;
}
void insqueue();
void delqueue();
void dispqueue();
};
```

```
void queue::insqueue()
{
node *ptr;
ptr=new node;
cout<<"Enter the element to be insert"<<endl;
cin>>ptr->data;
ptr->next=NULL;
if(rear==NULL)
 front=rear=ptr;
else
 {
 rear->next=ptr;
 rear=ptr;
 }
}
void queue ::delqueue()
{
 node *ptr;
 ptr=front;
 cout<<"The deleted element is "<<ptr->data;
 if(front==rear)
 front=rear=NULL;
 else
 front=front->next;
 delete ptr;
}
```

```
void queue :: dispqueue()
{
node *ptr;
ptr=front;
cout<<"The queue is "<<endl;
while(ptr!=NULL)
{
cout<<ptr->data<<endl;
ptr=ptr->next;
}
}
```

```
void main()
{
clrscr();
char ans;
queue q1;
do
{
q1.insqueue();
cout<<"wish to continue "<<endl;
cin>>ans;
}while(ans=='y');
q1.dispqueue();
cout<<"Press any key to delete an element
...."<<endl;
getch();
q1.delqueue();
q1.dispqueue();
getch();
}
```

Some Questions based on Board Examination on Linked stack & Linked Queue

Q1. Write a function in C++ to delete a node containing customer's information, from a dynamically allocated Queue of Customers implemented with the help of the following structure:

```
struct   Customer
{int   CNo;
            char CName[20];
            Customer *Link;
};
```

Ans:
```
struct Customer
{
        int   CNo;
        char CName[20];
        Customer *Link;
} *Front, *Rear, *ptr;
void DELETE()
{
    if(Front = = NULL)
        cout<<"\n Queue Underflow\n";
    else
    {
        ptr = Front;
        Front = Front→Link;
        delete ptr;
     }
 }
```

Q2. Write a function in C++ to delete a node containing Book's information, from a dynamically allocated Stack of Books implemented with the help of the following structure.

```
struct Book
{int BNo;
            char BName[20];
            Book *Next;
        };
```

Ans:
```
struct Book
{
        int BNo;
        char BName[20];
        Book *Next;
}*top, *ptr;
void POP()
{
if(top = = NULL)
        cout<<"\n Stack Underflow\n";
    else
    {
        ptr = top;
        top = top→Next;
        delete ptr;       }
 }
```

Q 3.Evaluate the postfix notaion of expression.
4, 10 , 5 , + , * , 15 , 3 ,/ , -

Sno.	Symbol	Stack
0		[
1	4	[4
2	10	[4,10
3	5	[4,10,5
4	+	[4 [4,15
5	*	[[60
6	15	[60,15
7	3	[60,15,3
8	/	[60 [60,5
9	-	[[55
10	]	55 **Ans**

Q. 4. Convert the following infix expression to its equivalent postfix expression, showing the stack contents for each step of conversion.

X / Y + U* (V-W)

Ans. :- X / Y + U* (V-W)=((X / Y)+(U*(V-W)))

Element	Stack	Postfix
(		
(		
X		X
/	/	X
Y	/	XY
)		XY/
+	+	XY/
(	+	XY/
U	+	XY/U
*	+*	XY/U
(	+*(	XY/U
V	+*(	XY/UV

	-	+*(-	XY/UV	
	W	+*(-	XY/UVW	
	)	+*	XY/UVW-	
X	)	+	XY/UVW-*	/ Y
+ U*	)		XY/UVW-*+	

(V-W)= XY/UVW-*+