EX.NO : 7(A)	STACK USING LINKED LIST
DATE :	

PROGRAM STATEMENT:

To write a CPP Program to insert five special character elements in to stack using linked list.

ALGORITHM:

1. Start the program/
2. Define a stack of type char to hold elements.
3. Input the number of elements (n) to be pushed onto the stack.
4. Use a loop to input n characters and push them onto the stack.
5. Output the current size of the stack using s.size().
6. Output the top element of the stack using s.top(), then pop the element and repeat for the second pop operation.
7. After two pop operations, output the new top element and the size of the stack.
8. End the program.

PROGRAM:

```
#include<iostream>
#include<stack>

usingnamespacestd;
int main()
{
  stack<char>
  s; int n;
  char data;
  cin>>n;
  for(int i=0;i<n;i++){
     cin>>data;
     s.push(data);
     }
      cout<<"Current size of the stack is "<<s.size()<<endl;
    cout<<"Thetopelement ofthe stack is "<<s.top()<<endl;
    s.pop();
  cout<<"Thetopelement after 1 popoperation is "<<s.top()<<endl;
  s.pop();
```

```
cout<<"The top element after 2 pop operations is "<<s.top()<<endl;
cout<<"Size ofthestack after 2 popoperations is "<<s.size()<<endl;

}
```

OUTPUT :

	Input	Expected	Got	
✔	5 @ # $ % ^	Current size of the stack is 5 The top element of the stack is ^ The top element after 1 pop operation is % The top element after 2 pop operations is $ Size of the stack after 2 pop operations is 3	Current size of the stack is 5 The top element of the stack is ^ The top element after 1 pop operation is % The top element after 2 pop operations is $ Size of the stack after 2 pop operations is 3	✔
✔	4 " . , :	Current size of the stack is 4 The top element of the stack is : The top element after 1 pop operation is , The top element after 2 pop operations is . Size of the stack after 2 pop operations is 2	Current size of the stack is 4 The top element of the stack is : The top element after 1 pop operation is , The top element after 2 pop operations is . Size of the stack after 2 pop operations is 2	✔

Passed all tests! ✔

RESULT:

Thus, the C++ program to insert five special character elements in to stack using linked list is created successfully.

EX.NO : 7(B)	STACK APPLICATION USING LINKED LIST
DATE :	

PROGRAM STATEMENT:

To write a CPP program for Infix to Prefix Conversion using Linked List Stack STL.

ALGORITHM:

1. Start the program.
2. Define a Node structure with a data field and a pointer to the next node. Initialize top as NULL.
3. Implement a push function to add elements to the stack, creating a new node, assigning it the input data, and setting its next pointer to the current top.
4. Implement a pop function to remove the top element from the stack, adjust the top pointer, and delete the popped node.
5. Implement an isEmpty function to check if the stack is empty.
6. Implement a precedence function to return the precedence value for the operators: 1 for +, -, 2 for *, /, and 3 for others (like parentheses).
7. Convert an infix expression to prefix using the stack by reversing the input expression, processing each character based on precedence, and constructing the prefix expression. Return the converted prefix expression.
8. End the program.

PROGRAM:

```
#include<bits/stdc++.h>
using namespace std;
#include<iostream>
#include<cstring>
struct Node{
char data;
struct Node*next;
}*top=NULL;

void push(char x)
{
struct Node *p = new Node; if(p==NULL)
{
p->data= x;
p->next =top;
top = p;
```

```
    }
    else
    {
     p->data= x;
    p->next =to
    continue;
    } else{
          while(!s.empty() && t<precedence(s.top())){
              prefix+=s.top(); s.pop();
            }
            s.push(infix[i]); continue;
         }
       }
    }
       while(!s.empty()){
          prefix      +=      s.top();
          s.pop();
       }
       reverse(prefix.begin(),prefix.end());
       returnprefix;
     }


    int main(){ char *infix = new
       char;
       cin>>infix;
       cout<<infixToPrefix(infix);
}
```

OUTPUT:

	Input	Expected	Got	
✔	((a+(b*c))-d)	-+((a*(bc))d)	-+((a*(bc))d)	✔
✔	A*(B+C)/D	+*A(B/C)D	+*A(B/C)D	✔
✔	((A*B)+(C/D))	+*((AB)/(CD))	+*((AB)/(CD))	✔

Passed all tests! ✔

RESULT:

Thus, the C++ program to Infix to Prefix Conversion using Linked List Stack STL is created successfully.

EX.NO : 7(C)	QUEUE USING LINKED LIST
DATE :	

PROGRAM STATEMENT:

To write a CPP Program to implement Queue using Linked List, insert integer elements in to Queue and delete two items from Queue.

ALGORITHM:

1. Start the program.
2. Define Node: Create a Node class with data and next pointer.
3. Push: Insert a new node at the rear of the queue.
4. Pop: Remove the front node.
5. Display: Print the queue from front to rear.
6. Display Front/Rear: Print front and rear node data.
7. Main: Perform pop(), push(), and display queue operations.
8. End the program.

PROGRAM:

```
#include<iostream>
using namespace std;

 class Node{
 public:
       double data;
        Node*next;
}*rear,*front,*temp;
void push(){
    temp = new Node;
    cin>>temp->data;
    temp->next=NULL;
    if(rear == NULL)
    {
    front =temp;
    rear = temp;
    }
    else
```

```
   {
  rear->next =temp;
   rear = rear->next;
   }
}
void pop(){
 if(rear == NULL)
{
 cout<<"Underflow."<<endl;
}
else
    {
    temp= front;
     front = front->next;
    temp->next =NULL;
    free(temp);
    }
}
void display()
{
temp= front;
while(temp!=NULL){
       cout<<temp->data<<" ";
        temp = temp->next;
       }
}
void disp(){
    temp= front;
    cout<<"Queue Front : "<<temp->data;
while(temp->next !=NULL)
{
temp = temp->next;
}
 cout<<endl;
cout<<"Queue Rear : "<<temp->data; }
int main()
{
int n; cin>>n;
pop();
for(inti=0;i<n;i++)
{
push();
}
cout<<"Queue :";
display();
```

```
    pop();
    pop();
    cout<<endl;
    cout<<"After DeQueue:";
    display();
    cout<<endl;
    disp();

}
```

OUTPUT :

	Input	Expected	Got	
✔	5 10 20 30 40 50	Underflow. Queue :10 20 30 40 50 After DeQueue :30 40 50 Queue Front : 30 Queue Rear : 50	Underflow. Queue :10 20 30 40 50 After DeQueue :30 40 50 Queue Front : 30 Queue Rear : 50	✔
✔	4 100 200 300 400	Underflow. Queue :100 200 300 400 After DeQueue :300 400 Queue Front : 300 Queue Rear : 400	Underflow. Queue :100 200 300 400 After DeQueue :300 400 Queue Front : 300 Queue Rear : 400	✔

Passed all tests! ✔

RESULT:

Thus, the C++ program to implement Queue using Linked List, insert integer elements in to Queue and delete two items from Queue is created successfully.

EX.NO : 7(D)	QUEUE APPLICATION USING LINKED LIST
DATE :	

PROGRAM STATEMENT:

To write a C++ program to implement FCFS algorithm(no of.process p1,p2 and p3 and its burst time are 10,5 & 8) find out waiting time of the process.

ALGORITHM:

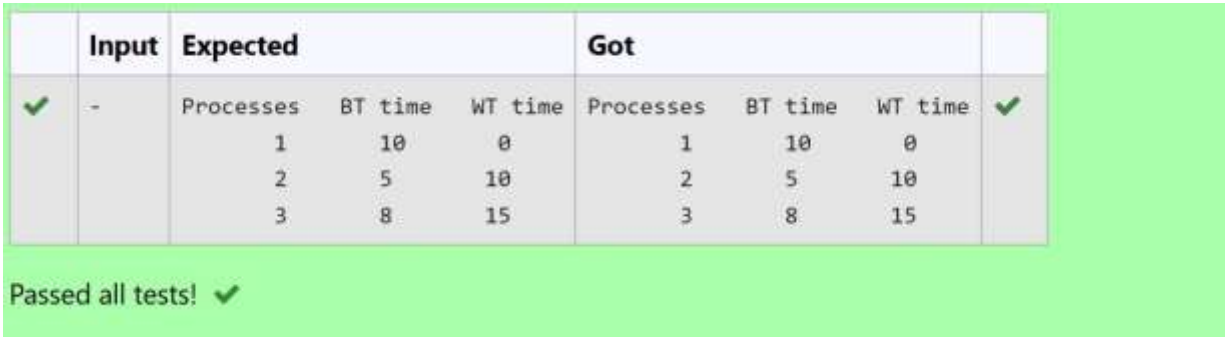
1. Start the program.
2. Initialize Variables: Define arrays for burst time (bt), waiting time (wt), and turn-around time (tat).
3. Input Burst Times: Set burst times for 3 processes (e.g., 10, 5, 8).
4. Calculate Waiting Times: For each process (except the first), sum the burst times of the previous processes.
5. Calculate Turnaround Times: Add burst time and waiting time for each process.
6. Display Results: Print process number, burst time, and waiting time for each process.
7. End the Program: No average calculation needed.

PROGRAM:

```
#include<iostream>
using namespace std;
int main()
{
  int bt[20], wt[20], tat[20], i, j, n, total=0;
  n=3;
  bt[0]=10,bt[1]=5,bt[2]=8;
  wt[0]=0;
  for(i=1;i<n;i++)
  {
  wt[i]=0;
  for(j=0;j<i;j++) wt[i]+=bt[j];
    total+=wt[i];
  }
  total=0;
  cout<<"Processes BT time WT time"<<endl;
  for(i=0;i<n;i++)
  {
  tat[i]=bt[i]+wt[i];
```

```
    total+=tat[i];
    cout<<"          "<<i+1<<"      "<<bt[i]<<"     "<<wt[i]<<endl;
    }
}
```

OUTPUT :



	Input	Expected			Got			
✔	-	Processes	BT time	WT time	Processes	BT time	WT time	✔
		1	10	0	1	10	0	
		2	5	10	2	5	10	
		3	8	15	3	8	15	

Passed all tests! ✔

RESULT:

Thus, the C++ program to implement FCFS algorithm (no of.process p1, p2 and p3 and its burst time are 10,5 & 8) find out waiting time of the process is created successfully.