EX.NO : 5(A)	STACK ADT
DATE :	

PROGRAM STATEMENT :

To Write a CPP Program to insert five character elements in to Stack ADT (use STL for Stack.

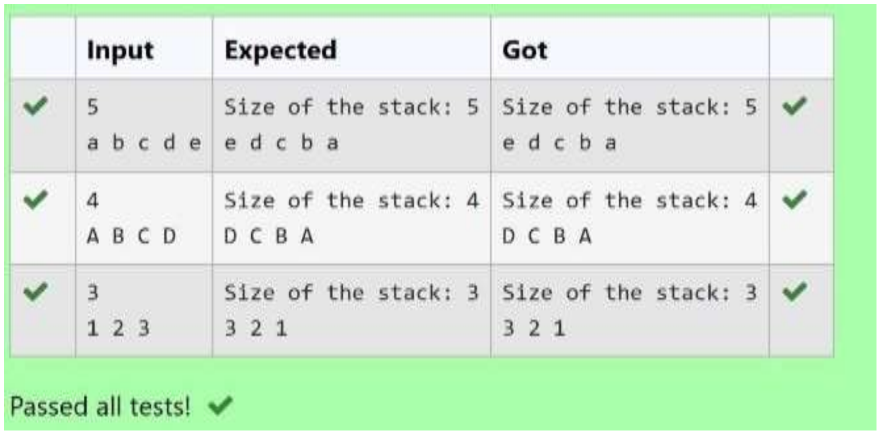
ALGORITHM:

1. Start the program.
2. Initialize a stack of characters named mystack.
3. Read an integer n as the size of the stack.
4. Print the size of the stack.
5. Use a loop to read n characters and push each character onto the stack.
6. Use a loop to pop and print each character from the stack until the stack is empty.
7. End the program.

PROGRAM:

```
#include<iostream>
#include<stack>
usingnamespacestd;
int main()
{
   stack<char>mystac
   k;
   int  n,i;
   char  c;
   cin>>n
   ;
   cout<<"Size ofthestack:
   "<<n<<endl; for(i=0;i<n;i++){
   cin>>c; mystack.push(c);
   }
   while(!mystack.empty()){
      cout<<mystack.top()<<" "; mystack.pop();
   }
}
```

OUTPUT :



	Input	Expected	Got	
✓	5 a b c d e	Size of the stack: 5 e d c b a	Size of the stack: 5 e d c b a	✓
✓	4 A B C D	Size of the stack: 4 D C B A	Size of the stack: 4 D C B A	✓
✓	3 1 2 3	Size of the stack: 3 3 2 1	Size of the stack: 3 3 2 1	✓

Passed all tests! ✓

RESULT :

Thus, the C++ program to insert five character elements in to Stack ADT is created successfully.

EX.NO : 5(B)	STACK APPLICATIONS
DATE :	

PROGRAM STATEMENT :

To write a CPP program for Infix to Postfix Conversion using Stack STL

ALGORITHM:

1. Define a function priority to assign precedence to operators.
2. Create a postfix string and a stack to hold operators.
3. Loop through each character in the infix expression, processing operands and operators.
4. Handle parentheses by pushing and popping from the stack.
5. After processing the expression, pop any remaining operators from the stack to postfix.
6. Print the resulting postfix expression.
7. End the program

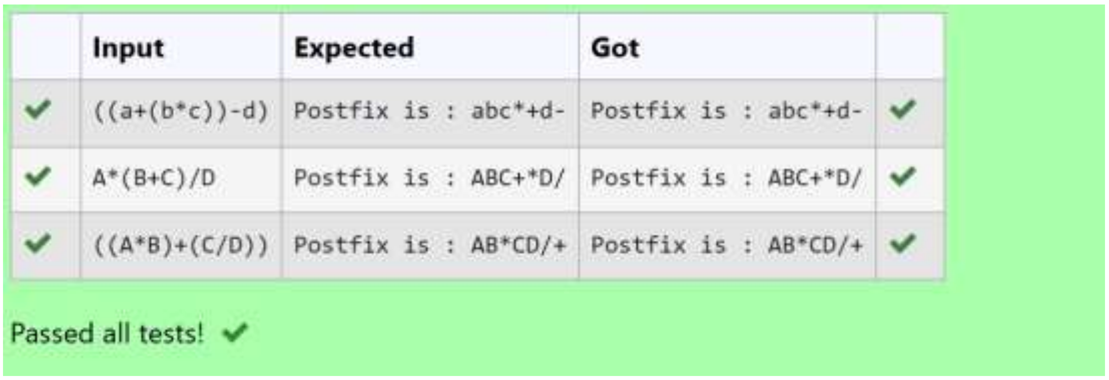
PROGRAM:

```
// Usinginbuilt stack libraryto createstack
#include <iostream>
#include <stack>
#include <string>
usingnamespacestd;

int priority (char alpha){ if(alpha == '+' ||alpha =='-') return 1;

  if(alpha  ==  '*'  ||  alpha=='/')
    return 2;

  if(alpha    ==    '^')
    return 3;


  return  0;
}
string convert(string infix)
{ int i= 0;
  stringpostfix = "";
```

```
// using inbuilt stack<> from C++stacklibrary

 stack<int> s;
while(infix[i]!='\0')
{
  // ifoperand add to thepostfix expression if( ( (infix[i]>='a') &&
  (infix[i]<='z') ) || ( (infix[i]>='A') && (infix[i]<='Z')) )
  { postfix+=
    infix[i]; i++;
  }
  // ifopening bracket thenpushthestack
  else if(infix[i]=='(')
  {
    s.push(infix[i]);
    i++;
  }
  // if closing bracket encounted thenkeep popping fromstack until
  // closingapairopeningbracket is not encountered
  else if(infix[i]==')')
  {
    while(s.top()!='(')
    {
      postfix      +=      s.top();
      s.pop();
    }
    s.pop()
  ; i++; }
  else
  { while(!s.empty() && priority(infix[i]) <= priority(s.top())){
      postfix += s.top(); s.pop();
    }
    s.push(infix[i]);
    i++;
  }
}
while(!s.empty()){
  postfix      +=      s.top();
  s.pop();
}


cout << "Postfix is : " << postfix;//it willprint postfixconversion
return postfix;
}
```

```
int main()
{ string
  infix;
  string
    postfix; cin>>infix; postfix == convert(infix);


    return 0;
 }
```

OUTPUT :



	Input	Expected	Got	
✓	((a+(b*c))-d)	Postfix is : abc*+d-	Postfix is : abc*+d-	✓
✓	A*(B+C)/D	Postfix is : ABC+*D/	Postfix is : ABC+*D/	✓
✓	((A*B)+(C/D))	Postfix is : AB*CD/+	Postfix is : AB*CD/+	✓

Passed all tests! ✓

RESULT :

Thus, the C++ program for Infix to Postfix Conversion using Stack STL is created successfully.

EX.NO : 5(C)	QUEUE ADT
DATE :	

PROGRAM STATEMENT :

To write a CPP Program to insert 5 operator elements in to Queue ADT.

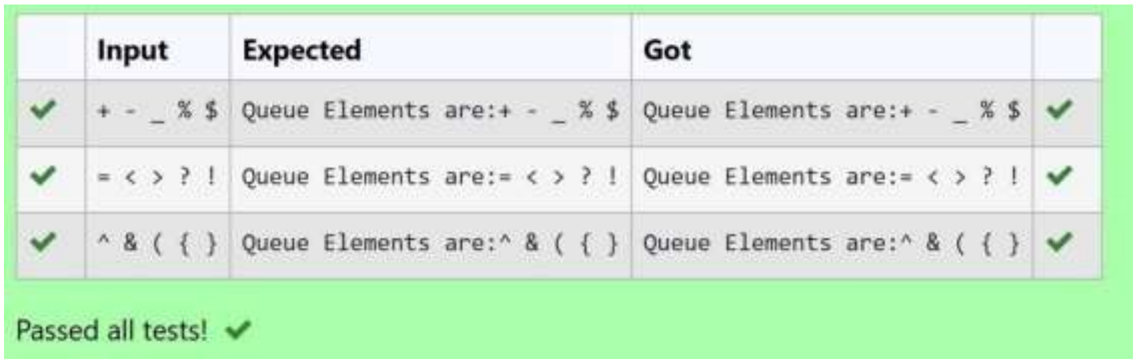
ALGORITHM:

1. Create a queue of characters using queue<char>.
2. Input 5 characters using a loop and push each character into the queue.
3. Print the message "Queue Elements are:".
4. Use a while loop to traverse the queue until it is empty.
5. For each iteration, print the front element of the queue using que.front().
6. Remove the front element from the queue using que.pop()
7. End the program

PROGRAM:

```
#include
<iostream>
#include  <queue>
usingnamespacest
d; int main()
{
  queue<char>qu
  e; char c;
  for(int i=0;i<5;i++)
  { cin>>c;
     que.push(c
     );
  }
  cout<<"QueueElementsare:"; while(!que.empty())
  { cout<<que.front()<<"
     "; que.pop();
  }
}
```

OUTPUT :



	Input	Expected	Got	
✓	`+ - _ % $`	`Queue Elements are:+ - _ % $`	`Queue Elements are:+ - _ % $`	✓
✓	`= < > ? !`	`Queue Elements are:= < > ? !`	`Queue Elements are:= < > ? !`	✓
✓	`^ & ( { }`	`Queue Elements are:^ & ( { }`	`Queue Elements are:^ & ( { }`	✓

Passed all tests! ✓

RESULT :

Thus, the C++ program to insert 5 operator elements in to Queue ADT is created successfully.

EX.NO : 5(D)	QUEUE APPLICATIONS
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 each process & Average waiting time of the process.

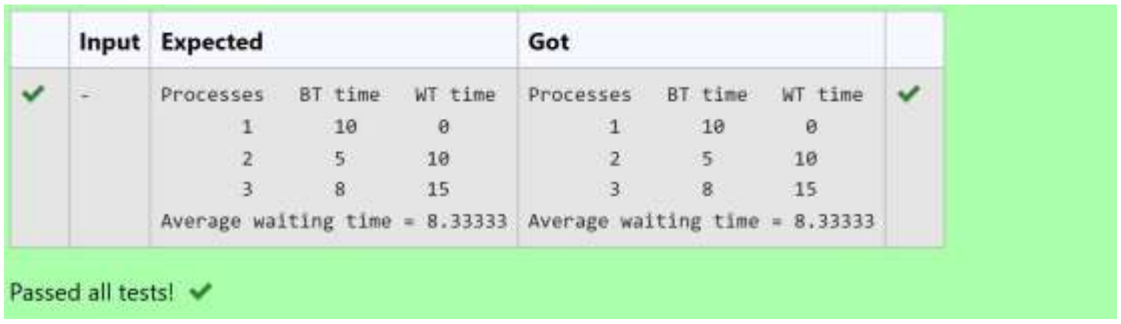
ALGORITHM:

1. Start the program.
2. Define findWaitingTime to take process IDs, count of processes n, and burst times bt.
3. Initialize wt for waiting times and set wt[0] = 0.
4. Use a loop to calculate each waiting time as wt[i] = bt[i-1] + wt[i-1].
5. Print "Processes", "BT time", and "WT time" table headings.
6. Use a loop to print each process's ID, burst time, and waiting time, adding each to total_wt.
7. Print average waiting time as total_wt / n.
8. In main, define processes, n, and bt, then call findWaitingTime.
9. End the program.

PROGRAM:

```
#include<iostream>
using namespace std;
void findWaitingTime(int processes[], int n,int bt[])
{ intwt[n]; float
  total_wt=0;
  // waitingtimefor first process is 0
  wt[0]=0;
  //
  calculatingwaitingtime
  for (int i= 1; i< n ; i++ )
  wt[i] = bt[i-1] +wt[i-1] ;


  cout << "Processes "<<" BT time "<< " WT time \n";
  for (int i=0; i<n; i++)
  {  total_wt  =total_wt  +
   wt[i];
    cout << "       " << i+1 << "  " << bt[i] <<"  "<<wt[i]<<endl; }
cout<<"Averagewaitingtime="<<total_wt/n<<endl; }
//  Driver  code
int main()
{
   //process   id's   int
processes[]={1,2,3};
int n = 3;

int bt[]={10,5,8};


   findWaitingTime(processes,   n,   bt);
   return 0;
}
```

OUTPUT :



	Input	Expected	Got	

✓	-	
		```
		Processes   BT time   WT time
		        1        10         0
		        2         5        10
		        3         8        15
		Average waiting time = 8.33333
		```
			```
			Processes   BT time   WT time
			        1        10         0
			        2         5        10
			        3         8        15
			Average waiting time = 8.33333
			```
				✓

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 each process & Average waiting time of the process ADT is created successfully.