55. Convert the following infix expression to its equivalent postfix expression, showing the stack contents for each step of conversion.
X / Y + U* (VW)

Ans 55. X / Y + U* (VW)=((X / Y)+(U*(VW)))

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-
)	+	XY/UVW-*
)		XY/UVW-*+

56. Convert the following infix expression to its equivalent Postfix expression, showing the stack contents for each step of conversion.
U * V + R/ (ST)

ANs

56.

OR

Element	Stack	Postfix
U		U
*	*	U
V	*	UV
+	+	UV*
R	+	UV*R
/	+/	UV*R
(	+/(	UV*R
S	+/(	UV*RS
-	+/(-	UV*RS
T	+/(-	UV*RST
)	+/	UV*RST-
	+	UV*RST-/
		UV*RST-/+

57. Translate, following infix expression into its equivalent postfix expression: ((A-B)*(D/E))/(F*G*H)

Ans 57. Equivalent postfix expression:

=((A-B)*(D/E))/(F*G*H)

=((AB-)*(DE/))/(FG*H*)

=AB – DE /* FG* H*/

58. Translate, following infix expression into its equivalent postfix expression: A*(B+D)/E-F-(G+H/K)

Ans 58. Equivalent postfix expression:

= A*(B+D)/E-F-(G+H/K)
= (A*(B+D)/E) – (F – (G + (H/K)))
= (A*(BD+)/E) – (F – (G + (HK/)))
= ((ABD+*) / E) – (F – (GHK/+))
= ABD+* E/F – GHK/+ -

59. Write the equivalent infix expression for 10,3,*,7,1,-,*,23,+

Ans 59. 10 * 3 * (7 – 1) + 23

60. Write the equivalent infix expression for a, b, AND, a, c, AND, OR.

Ans 60. a, b, AND, a, c, AND, OR
(a AND b), (a AND c), OR
(a AND b) OR (a AND c)

61. Evaluate the infix expression.
P: 12 , 7 , 3 , - , / , 2 , 1 , 5 , + , * , + ,)

Ans 61.

Symbol	Stack
12	12
7	12,7
3	12,7,3
-	12,4
/	3
2	3,2
1	3,2,1
5	3,2,1,5
+	3,2,6
*	3,12
+	15
)	15

62. Give postfix form of the following expression A*(B+(C+D)*(E+F)/G)*H

Ans 62. A*(B+(CD+EF+*)/G)*H
A*(B+CD+EF+*G/))*H
(A*(BCD+EF+*G/+))H
(ABCD+EF+*G/+*)*H
ABCD+EF+*G/+*H*

63. Give postfix form expression for: NOT A OR NOT B AND NOT C

Ans 63. =((A NOT) OR ((B NOT) AND (C NOT)))
=((A NOT) OR ((B NOT C NOT AND))
=A NOT B NOT C NOT AND OR

64. Consider the infix expression Q : A+B * C ↑ (D/E)/F.
Translate Q into P, where P is the postfix equivalent expression of Q. what will be the result of Q if this expression is evaluated for A, B, C, D, E, F as 2, 3, 2, 7, 2, 2 respectively.

Ans 64. P = ABCDE/^*F/+
2,3,2,7,2
2,3,2->7/2->3

2,3,2,3
2,3->2^3->8
2,3,8
2->3*8->24
2,24,2
2->24/2->12
2,12
2+12
Result of evaluation = 14

65. Change the following infix expression into postfix expression.
(A+B) *C+D/E-F

Ans 65. Equivalent postfix expression:
= (A+B) *C+D/E-F
= (((A+B)*C) + (D/E)) – F
= (((AB+)*C) + (DE/)) – F
= AB+ C* DE/ + F-

66. Evaluate the following postfix expression. Show the status of stack after execution of each operation separately;
F, T, NOT, AND, F, OR, T, AND

Ans 66. F, T, NOT, AND, F, OR, T, AND

Scanned Element	**Operation**	**Stack Status**
F	Push	F
T	Push	F, T
NOT	Pop one operand from stack	F
	NOT T = F	
	Push	F, F
AND	Pop two operands from stack	
	F AND F = F	
	Push	F
F	Push	F, F
OR	Pop two operands from stack	
	F OR F = F	
	Push	F
T	Push	F, T
AND	Pop two operands from stack	
	F AND T = F	
	Push	
	Pop all	F
Result	F	

67. Evaluate the following postfix expression. Show the status of stack after execution of each operation separately;
T, F, NOT, AND, T, OR, F, AND

Ans 67. T, F, NOT, AND, T, OR, F, AND

Scanned Element	**Operation**	**Stack Status**
T	Push	T
F	Push	T, F
NOT	Pop one operand from stack	T
	NOT F = T	
	Push	T, T

AND	Pop two operands from stack	
	T AND T = T	
	Push	T
T	Push	T, T
OR	Pop two operands from stack	
	T OR T = T	
	Push	T
F	Push	T, F
AND	Pop two operands from stack	
	T AND F = F	
	Push	F

Result F

68. Evaluate the following postfix expression. Show the status of stack after execution of each operation:

5, 2, *, 50, 5, /, 5, –, +

Ans 68. 5, 2, *, 50, 5, /, 5, –, +

Scanned Element	**Stack Status**
5	5
2	5.2
*	10
50	10, 50
5	10, 50, 5
/	10, 10
5	10, 10, 5
–	10, 5
+	15

69. Evaluate the following postfix expression. Show the status if stack after execution of each operation;

60, 6, /, 5, 2, *, 5, –, +

Ans 69. 60, 6, /, 5, 2, *, 5, –, +

Scanned Element	**Stack Status**
60	60
6	60, 6
/	10
5	10, 5
2	10, 5, 2
*	10, 10
5	10, 10, 5
–	10, 5
+	15

70. Evaluate the following postfix expression using a stack and show the contents of stack after execution of each operation;

5, 3, 2, *, 4, 2, /, –, *

Ans 70. 5, 3, 2, *, 4, 2, /, –, *

Scanned Element	**Stack Status**
5	5
3	5, 3

2	5, 3, 2
*	5, 6
4	5, 6, 4
2	5, 6, 4, 2
/	5, 6, 2
	5, 4
*	20

71. Evaluate the following postfix notation. Show status of stack after every step of evaluation (i.e. after each operator):
False, NOT, True, AND, True, False, OR, AND

Ans 71. False, NOT, True, AND, True, False, OR, AND

Scanned Element	**Operation**	**Stack Status**
False	Push onto stack.	False
NOT	Pop one element from the stack i.e. False. Apply NOT on	
False	we get true, push onto stack.	
True	Push onto stack.	True, True
AND	Pop two elements from the stack. Apply AND between them. Push result onto stack.	
True	Push True onto stack	True, True
False	Push False onto stack.	True, True, False
OR	Pop two elements from the stack. Apply OR between them and push result onto stack.	True, True
AND	Pop two elements from the stack. Apply and between them and push result onto stack.	True

72. Evaluate the following postfix notation, show status of stack of every step of evaluation (i.e. after each Operator):
True, False, NOT, AND, False, True, OR, AND

Ans 72. True, False, NOT, AND, False, True, OR, AND

Scanned Element	**Operation**	**Stack Status**
True	Push onto stack.	True
NOT	Pop one element from the stack. Apply NOT on False and push onto stack.	True, False
AND	Pop two elements from the stack. Apply AND between them. Push the result onto stack.	True
False	Push False onto stack.	True, False
True	Push True onto stack	True, False, True
OR	Pop two elements from the stack. Apply OR between them and push result onto stack.	True, True
AND	Pop two elements from the stack. Apply and between them	True

and push result onto stack.

73. Evaluate the postfix notation of expression:
50, 60, +, 20, 10, –, *

Ans 73. 50, 60, +, 20, 10, –, *

Scanned Element	**Stack Status**
50	50
60	50, 60
+	110
20	110, 20
10	110, 20, 10
–	110, 10
*	1100
	Pop all

74. Evaluate the following postfix notation of expression:
True, False, NOT, AND, True, True, AND, OR

Ans 74. True, False, NOT, AND, True, True, AND, OR

Scanned Element	**Stack Status**
True	True
False	True, False
NOT	True, True
AND	True
True	True, True
True	True, True, True
AND	True, True
OR	True

75. Evaluate the following postfix notation of expression:
False, True, NOT, OR True, False, AND, OR

Ans 75. False, True, NOT, OR True, False, AND, OR

Scanned Element	**Stack Status**
False	False
True	False, True
NOT	False, False
OR	False
True	False, True
False	False, True, False
AND	False, False
OR	False

76. Evaluate the following postfix notation of expression. Show the status of stack after each expression:
True, False, NOT, OR, False, True, OR, AND

Ans 76. True, False, NOT, OR, False, True, OR, AND

Scanned Element	**Stack Status**
True	True
False	True, False
NOT	True, True
OR	True
False	True, False
False	True, False

```
          True           True, False, True
          OR             True, True
          AND            True
```

77. Convert the following infix expression to its equivalent postfix expression. Showing stack contents for the conversion:
(X – Y / (Z + U)* V)

Ans 77. Let us rewrite like (X – Y / (Z + U) * V

Scanned Element	Stack Status	Expression
(	(	
X	(	X
–	(–	X
Y	(–	XY
/	(	

4 Marks Questions

78. Write the definition of a member function Pop() in C++, to delete a book from a dynamic stack of TEXTBOOKS considering the following code is already included in the program.

```
struct TEXTBOOKS
{
char ISBN[20]; char TITLE[80];
TEXTBOOKS *Link;
};
class STACK
{
TEXTBOOKS *Top;
public:
STACK() {Top=NULL;}
void Push();
void Pop();
~STACK();
};
```

Ans 78.

```
void STACK::POP()
{
if (Top!=NULL)
{
TEXTBOOKS *Temp;
Temp=Top;
cout<<Top->ISBN<<Top->TITLE<<"deleted"<<endl;
Top=Top>
Link;
delete Temp;
}
else
cout<<"Stack Empty"<<endl;
}
```

79. Write the definition of a member function PUSH() in C++, to add a new book in a dynamic stack of BOOKS considering the following code is already included in the program:

```
struct BOOKS
```

```
{
char ISBN[20], TITLE[80];
BOOKS *Link;
};
class STACK
{
BOOKS *Top;
public:
STACK()
{Top=NULL;}
void PUSH();
void POP();
~STACK();
};
```

Ans 79.

```
void STACK::PUSH()
{
BOOKS *Temp;
Temp=new BOOKS;
gets(Temp->ISBN);
gets(Temp->TITLE);
Temp->Link=Top;
Top=Temp;
}
```

80. Write a complete program in c++ to implement a dynamically allocated Stack containing names of Countries.

Ans 80.

```
#include<iostream.h>
#include<stdio.h>
struct Node
{       char Country [20] ; Node *Link; };
class Stack
{ Node *Top;
public:
Stack( )
{ Top = NULL; }
void Push() ;
void Pop() ;
void Display() ;
~Stack () ;
};
void Stack::Push( )
{
Node *Temp = new Node;
gets(Temp -> Country);
Temp -> Link = Top;
Top = Temp;
}
void Stack::Pop( )
{
if (Top !=NULL)
{
Node *Temp = Top;
Top = Top -> Link;
```

```
delete Temp;
}
else
cout<<“stack Empty”;
}
void Stack::Display( )
{
Node *Temp = Top;
while (Temp! = NULL)
{
cout<<Temp -> Country <<endl;
Temp = Temp -> Link;
}
}
Stack::~Stack ( )
{
while (Top!=NULL)
{ NODE *Temp=Top;
Top=Top->Link;
delete Temp;
}
}
void main ( )
{ Stack ST;
char Ch;
do
{ cout<<“p/O/D/Q” ;
cin>>Ch;
switch (Ch)
{
case ‘P’ : ST.Push( ); break;
case ‘O’ :ST.Pop(); break;
case ‘D’ :ST.Disp();
}
} while (Ch!=’Q’);
}
```

81. Write a complete program in C++ to implement a dynamically allocated Queue containing names of Cities.

Ans 81.

```
#include <iostream.h>
#include <conio.h>

struct NODE
{ char City[20];
NODE *Next;
};
class Queue
{ NODE *Rear,*Front;
puplic:
Queue( )
{ Rear=NULL;Front=NULL;
}
void Qinsert( );
void Qdelete( );
void Qdisplay( );
```

```
~Queue( );
} ;
void Queue::Qinsert( )
{
NODE *Temp;
Temp=new NODE;
cout<<"Data:";
gets (Temp->City);
Temp->Next=NULL;
if (Rear==NULL)
{
Rear=Temp;
Front=Temp;
}
else
{
Rear->Next=Temp;
Rear=Temp;
}
}
void Queue::Qdelete( )
{
if (Front!=NULL)
{
NODE *Temp=Front;
cout<<Front->City<<"Deleted \n";
Front=Front->Next;
delete Temp;
if (Front==NULL)
Rear=NULL;
}
else
cout<<"Queue Empty..";
}
Queue:: Qdisplay( )
{ NODE *Temp=Front;
while (Temp!=NULL)
{
cout<<Temp->City<<endl;
Temp=Temp->Next;
}
}
Queue:: ~Queue( )//Destructor Function
{ while (Front!=NULL)
{ NODE *Temp=Front;
     Front=Front->Next; delete Temp;

}
}
void main( )
{ Queue QU;
char Ch;
do
```

```
     {
     :
     :
     } while (Ch!='Q');
          }
```

82. Write a function QUEINS( ) in C++ to insert an element in a dynamically allocated Queue containing nodes of the following given structure:

```
struct Node
{ int PId ; //Product Id
char Pname [20] ;
NODE *Next ;
} ;
```

ANs 82.

```
     void QUEINS (Node *&Front, Node *&Rear)
     { Node *Temp = new Node;
     cin>>Temp->PId;
     gets (Temp->Pname);
     //or cin>>Temp->Pname;
     //cin.get1ine(Temp->Pname);
     Temp->Next = NULL;
     if(Rear == NULL)
     Front = Temp;
     e1se
     Rear -> Next = Temp;
     Rear = Temp;
     }
```

83. Write a function QUEDEL( ) in C++ to display and delete an element from a dynamically allocated Queue containing nodes of the following given structure:

```
     struct NODE
     {
     int Itemno;
     char Itemname[20];
     NODE *Link;
     } ;
```

Ans 83.

```
     class Queue
     { Node *Front, *Rear;
     public:
     QUEUE( )//Constructor to initialize Front and Rear
     {
     Front = NULL;
     Rear = NULL;
     }
     void QUEINS( );      //Function to insert a node
     void QUEDEL( );      //Function to delete a node
     void QUEDISP( );     //Function to display nodes
     ~Queue();            //Destructor to delete all nodes
     };
     void Queue::QUEDEL( )
     { if (Front!=NULL)
     {NODE *Temp=Front;
```

```
cout<<Front->Itemno<<" ";
cout<<Front->Itemname<<"Deleted";
Front=Front->Link;
delete Temp;
if (Front NULL)
Rear=NULL;
}
else
cout<<"Queue Empty..";
}
```

84. 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 84.

```
struct Book
{
int BNo ;
char BName[20] ;
Book *Next ;
} ;
class Stack
{
Book *Top;
public:
Stack( )
{
Top = NULL;
}
void Push( );
void Pop( );
void Display( );
};
void Stack::Pop( )
{
Book *Temp;
if( Top= = NULL)
cout<<"Stack Underflow…";
else
{
cout<<"\nThe Book number of the
element to delete: "<<Top->BNo;
cout<<"\nThe Book name of the
element to delete: "<<Top->BName;
Temp=Top;
Top=Top->Next;
delete Temp;
}
```

```
}
```

85. Write a function in C++ to perform Insert operation in dynamically allocated Queue containing names of students.

```
Struct NODE
{ char Name[20];
NODE *Link;
};
```

Ans 85.

```
class Queue
{   NODE *front,*rear;
public:
Queue( )
{   front = rear = NULL;     }
void Insert( );
void Delete( );
void Display( );
};
void Queue::Insert( )
{
NODE *ptr;
ptr=new NODE;
if(ptr= = NULL)
{   cout<<"\nNo memory to create a
new node….";
exit(1);
}
cout<<"\nEnter the name….";
gets(ptr->Name);
ptr->Link=NULL;
if(rear= = NULL)
front=rear=ptr;
else
{
Rear->Link=ptr;
rear=ptr;
}
}
```

86. Write a function in C++ to perform a PUSH operation in a dynamically allocated stack considering the following :

```
struct Node
{
int X,Y ;
Node *Link ;
} ;
class STACK
{
Node *Top ;
public :
STACK( )
{Top = Null ;}
```

```
     void PUSH( ) ;
     void POP( ) ;
     ~STACK( ) ;
     } ;
```
ANs 86.
```
     struct Node
     { int X,Y ;
     Node *Link ;
     } ;
     class STACK
     { Node *Top ;
     public :
     STACK( )
     { Top = NULL;
     }
     void PUSH( ) ;
     void POP( ) ;
     ~STACK( ) ;
     } ;
     void STACK::PUSH( )
     { Node *Temp;
     Temp=new Node;
     if(Temp= =NULL)
     { cout<<"\nNo memory to create the node…";
     exit(1);
     }cout<<"Enter the value of X and Y";
     cin>>Temp->X>>Temp->Y;
     Temp->Link=Top;
     Top=Temp;
     }
```

87. Write a function QINSERT () in C++ to perform insert operation on a Linked Queue which contains client no and client name. Consider the following definition of NODE in the code of QINSERT()

```
  struct Node
     {  long int Cno;     //Client No
     char Cname[20];   //Clieht Name
     NODE *Next;

     };
```
Ans 87.
```
void QINSERT()
{  NODE *P=new NODE();
 cout<<"Enter the client number";
 cin>>P->Cno;
 cout<<"enter the client name";
 gets(P->Cname);
 P->Next=NULL;
 if(front==NULL && rear==NULL)
 { front=P;
  rear=P;
  }
  else
  { rear->Next=P;   rear=P;   }  }
```

88. Write a function PUSHBOOK() in C++ to perform insert operation on a Dynamic Stack, which contains Book_no and Book_Title. Consider the following definition of NODE, while writing your C++ code.

```
struct NODE
 { int Book_No;
 char Book_Title[20];
  NODE *Next;
};
```

Ans 88.
```
void PUSHBOOK(NODE *top)
{ NODE *NEW=new NODE;
  cout<<"Enter the book number";
  cin>>NEW->Book_No;
  cout<<"Enter book title";
  gets(NEW->Book_Title);
  NEW->Next=NULL;
  if(top==NULL)
  top=NEW;
  else { NEW->Next=top; top=NEW;}
  }
```

89. Write a function POPBOOK() in C++ to perform delete operation on a Dynamic Stack, which contains Book_no and Book_Title. Consider the following definition of NODE, while writing your C++ code.

```
struct NODE
 { int Book_No;
 char Book_Title[20];
  NODE *Link;
};
```

Ans 89.
```
void POPBOOK(NODE *top)
{
  cout<<"deleting top element from stack\n";
  cout<<"Book No"<<top->Book_No<<endl;
  cout<<"Book title"<<top->Book_Title<<endl;
  NODE *temp=top;
  top=top->Link;
  delete(temp);
  }
```

90. Write a function in C++ to perform a DELETE operation in a dynamically allocated queue considering the following description:

```
struct Node
{         float U,V;
     Node *Link;
};
class QUEUE
{    Node *Rear, *Front;
public:
     QUEUE( ) { Rear =NULL; Front= NULL;}
     void INSERT ( );
     void DELETE ( );
```

```
          ~QUEUE ( );
     };
```

Ans 90.

```
     void DELETE()
     {    Node *temp;
          if(front==NULL)                    // No element in the queue
          {    cout<<"UNDERFLOW……..";
          }
          else
          {    temp=front;
          front=front->Link;// Making the second node as the first one
               temp->Link = NULL;
               delete temp;   // deleting the previous first node.
          }
     }
```

91. Define stackpop( ) to delete nodes, for a linked list implemented stack having the following structure for each node:

```
     struct Node
     {    char name[20];
          int age;
          Node *Link;
     };
     class STACK
     {    Node * Top;
     public:
          STACK( ) { TOP=NULL;}
          void stackpush( );
          void stackpop( );
          ~STACK( );
     };
```

Ans 91.

```
     void stackpop( )                        // Pop from the beginning
     {    Node *temp;
          if(top==NULL)
          {    cout<<"UNDERFLOW…….";
          }
          else
          {    temp=Top;
               Top=Top->Link;
               temp->Link = NULL;
               delete temp;
          }}
```

92. Write an algorithm to insert an element from a linked queue depending upon user's choice.

Ans 92.
Algorithm for user choice:
1. ch=0 // Initialize a variable for user choice

2. Read ch //Enter, user choice 1 for push and 2 for pop
3. If(ch == 1) then /* if top is at end of Array-Queue */
4. call insert function
5. Else
6. call delete function
7. End

Algorithm for inserting in Linked-Queue:

/* Allocate space for ITEM to be inserted */
1. NEWPTR = new node
2. NEWPTR -> INFO = ITEM; NEWPTR -> LINK=NULL
/* Insert in the queue */
3. If rear = NULL Then
{
4. front = NEWPTR
5. rear = NEWPTR
6. else
{
7. rear -> LINK = NEWPTR
8. rear=NEWPTR
}
9. END.

93. Write an algorithm to delete an element from a linked queue depending upon user's choice.

Ans 93.

Algorithm for user choice:

1. ch=0 // Initialize a variable for user choice
2. Read ch //Enter, user choice 1 for push and 2 for pop
3. If(ch == 1) then /* if top is at end of Array-Queue */
4. call insert function
5. Else
6. call delete function
7. End

Algorithm for deleting in Linked-Queue:

1. If front==NULL Then
2. Print "Queue Empty"
3. Else
{
4. ITEM=front->INFO
5. If front=rear Then
{
6. front=rear=NULL
}
7. Else
8. front = front + 1
}
9. END

94. Write an algorithm that either pushes or pops an element in a linked stack, depending upon user's choice.

ANs 94.

Algorithm for user choice:

1. ch=0 // Initialize a variable for user choice
2. Read ch //Enter, user choice 1 for push and 2 for pop
3. If(ch == 1) then /* if top is at end of stack-array */
4. call push function
5. Else
6. call pop function
7. End

Algorithm for pushing in Linked-Stack

Step-1: If the Stack is empty go to step-2 or else go to step-3
Step-2: Create the new element and make your "stack" and "top" pointers point to it and quit.
Step-3: Create the new element and make the last (top most) element of the stack to point to it
Step-4: Make that new element your TOP most element by making the "top" pointer point to it.

Algorithm for popping in Linked-Stack:

Step-1: If the Stack is empty then give an alert message "Stack Underflow" and quit; or else proceed
Step-2: If there is only one element left go to step-3 or else step-4
Step-3: Free that element and make the "stack", "top" and "bottom" pointers point to NULL and quit
Step-4: Make "target" point to just one element before the TOP; free the TOP most element; make "target" as your TOP most element

95. Write an algorithm to insert an element in a circular queue implemented as an array.

Ans 95.

Let Q be a Queue having size N. DATA be the element to be inserted. F and R denote front and rear positions in the queue.

1. If (R=N-1) then R=0
   else
   R=R+1
2. If (F=R)
   { write ('Queue overflow')
     Goto step 5
   }
3. Q[R]=DATA
4. If (F=-1)then
   F=F+1
5. End

96. Write an algorithm to deleting an element from a circular queue implemented as an array.

Ans 96.

Let Q be a Queue having size N. DATA be the temporary variable which stores the deleted element (if possible). F and R denote front and rear positions in the queue.

```
1. If (F<0) then
   {
    write("Queue Underflow")
     goto step 4
   }
2. DATA=Q[F]
3. If(F=R) then
   { F=R=-1 }
   else
   {
    if(F=N-1)
    F=0
    else
    F=F+1
   }
4. End
```