1. Write a program to stimulate the working of stack using an array with the following:

(a) Push()
(b) Pop()
(c) Display

A:

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 5
void push(int);
void pop();
void display();
int stack[SIZE], top=-1;

void pop()
 {
  if(top==-1)
  {
  printf("Stack is empty!");
  }
  else
  {
  printf("The deleted element is %d, stack[top]);
  top=top-1;
  }
}
void push(int)
 {
 if(top==SIZE-1)
  {
   printf("Overflow");
  }
```

```
else
 {
 top = top + 1;
 stack[top] = value;
 printf ("Insertion successful");
  }
}
void display ()
 {
 int i;
 if (top == -1)
  {
  printf ("\nstack is empty");
  }
 for(i=top; i>=0; i--)
  {
  printf (" %d", stack[i]);
  }
}
int main () {
 int value, choice;
 while (1)
 {
 printf ("\n 1. Pop 2. Push \n 3. Display \n 4. Exit");
 scanf (" %d", &choice);
 switch (choice)
  {
 case 1: POP():
        break;
 case 2: printf ("Enter a value");
        scanf (" %d", &value);
        push (value);
        break;
```

```
case 3: display ( )
        break;
case 4: exit (0);
default: printf ("Wrong input value");
```

OUTPUT :

1. POP
2. PUSH
3. DISPLAY
4. EXIT

2

Enter a value : 6

Insertion successful.

2.

The deleted element is : 6

3.

Stack is empty.

```
#include <stdio.h>
#include <stdlib.h>
int index1=0, pos=0, top=-1, length;
char symbol, Temp, infix[20], postfix[20], stack[20];
void infixtopostfix();
void push(char symbol);
char(pop);
int pred (char symb);
void main()
 {
printf("Enter infix expression : \n");
scanf("%s", infix);
infixtopostfix();
printf("\n Intfix expression : \n %s", infix);
printf("\n Postfix expression: \n %s", postfix);
 }

void infixtopostfix()
 {
length = strlen(infix);
push('#');
while (index1 < length)
 {
symbol = infix[index1];
switch (symbol)
{
case '(': push(symbol);
          break;
```

```
        postfix[pos] = temp;
        pos ++;
        temp = pop();
        }
        break;
case '+':
    case '-':
case '*':
case '/':
case '^': while (pred (stack[top]) >= pred(symbol))
        {
        temp = pop();
        postfix[pos++] = temp;
        }
        push (symbol);
        break;
default : postfix [pos ++] = symbol;
    }
    index 1 ++;
}
while (top > 0)
    {
    temp= pop();
    postfix [pos ++] = temp;
    }
}
void push (char symbol)
    {
    top = top + 1;
    stack [top] = symbol;
    }
```

```
char pop()
{
char symb;
symb = stack[top];
top = top - 1;
return(symb);
}
int pred(char symbol)
{
int p;
switch(symbol())
{
 case '^': p=3;
break;
 case '*':
case '/': p=2;
   break;
case '+':
case '-': p=1;
    break;
 case '(': p=0;
    break;
 case '#': p=-1;
   break;
}
return(p);
}
```

Op: Enter the infix expression : A+B
postfix expression : AB+.

Op: Enter
1. Push
2. Pop
3. Disp
4. Exit
choice

Enter a

QUEUES
1.

OUTPUT
Operation
1. Insert
2. Delet
3. Displa
choice
the n
→ The op
1. Inse
2. Del
3. Di
cho
4
→ The
1.
2
3.
3.

o/p: Enter operation:
1. Push
2. Pop
3. Display
4. Exit
choice: 1

Enter a value 6

QUEUES OUTPUT

1. Operation to be performed.
1: Insert
2. Delete
3. Display
choice: 1
the no. to be inserted: 4
→ The operation to be performed
1. Insert
2. Delete
3. Display
choice: 2
4 is deleted
→ The operation to be performed
1. Insert
2. Delete
3. Display
3. Display