Experiment No 3
Problem Statements

1.Write a program to implement stack using array.

```
//1.Write a program to implement stack using array


/*Name:- Anuj Rajendra Mane
ROll No:- 65
Div:-A
Subject:- Data Structures*/


#include<stdio.h>
#include<stdlib.h>
int stack[5],top=-1;
void push() {
   int item;
   if(top==5-1)
   {
      printf("Stack is full:\n");
   }
   else
   {
      printf("Enter Push element in stack\n");
      scanf("%d",&item);
      top=top+1;
      stack[top] = item;
   }
}
void pop() {
   if (top==-1)
   {
      printf("Stack is Empty:\n");
   }
   else
   {
      printf("Popped %d\n",stack[top]);
      top=top-1;
   }
}
void show() {
   int i;
   if(top>=0)
   {
      printf("Stack Element is:\n");
      for(i=top;i>=0;i--)
      {
         printf("%d ",stack[i]);
      }
   }
   else
```

```
    {
        printf("Stack is empty\n");
    }
}
void peek() {
    if(top==-1)
    {
        printf("Stack is empty\n");
    }
    else
    {
        printf("Peek Element
is:%d\n",stack[top]);
    }
}
int main() {
    int ch;
    printf("Enter 1.For Push\n");
    printf("Enter 2.For Pop\n");
    printf("Enter 3.For Peek\n");
    printf("Enter 4.For Show\n");
    printf("Enter 5.For Exit\n");
    while(1) {
        printf("Enter Choice:\n");
        scanf("%d",&ch);

        switch(ch) {
            case 1:push();
            break;
            case 2:pop();
            break;
            case 3:peek();
            break;
            case 4:show();
            break;
            case 5:exit(0);
            break;
            default:
            printf("Invalid Option\n");
        }
    }
}
```

2. Write a program to convert a given infix expression to postfix form using stacks.

```
//2. Write a program to convert a
given infix expression to postfix
form using stacks.

/*Name:- Anuj Rajendra Mane
ROll No:- 65
Div:-A
Subject:- Data Structures*/
```

```
#include <stdio.h>
#include<stdlib.h>
#include <ctype.h>
char stack[20];
int top = -1;
void push(char x)
{
   stack[++top] = x;
}
char pop()
{
   if (top == -1)
      return -1;
   else
      return stack[top--];
}
int priority(char x)
{
   if (x == '(')
      return 0;
   if (x == '+' || x == '-')
      return 1;
   if (x == '*' || x == '/')
      return 2;
}
```

```
int main()
{
   char exp[20];
   char *e, x;
   printf("Enter the expression :: ");
   scanf("%s", exp);
   e = exp;
   while (*e != '\0')
   {
      if (isalnum(*e))
         printf("%c", *e);
      else if (*e == '(')
         push(*e);
      else if (*e == ')')
      {
         while ((x = pop()) != '(')
            printf("%c", x);
      }
      else
      {
         while (priority(stack[top]) >=
priority(*e))
            printf("%c", pop());
         push(*e);
      }
      e++;
   }
```

```
    while (top != -1)
    {
        printf("%c", pop());
    }
}
```

3. Write a program evaluating a postfix expression using stack.

```
//3.Write a program evaluating a postfix expression using stack.
/*Name:- Anuj Rajendra Mane
ROll No:- 65
Div:-A
Subject:- Data Structures*/

#include<stdio.h>
#include<ctype.h>
int stack[20];
int top = -1;

void push(int x)
{
    stack[++top] = x;
}
int pop()
{
    return stack[top--];
}
int main()
{
    char exp[20];
    char *e;
    int n1,n2,n3,num;
    printf("Enter the expression :: ");
    scanf("%s",exp);
    e = exp;
    while(*e != '\0')
    {
        if(isdigit(*e))
        {
            num = *e - 48;
            push(num);
        }
        else
        {
            n1 = pop();
            n2 = pop();
            switch(*e)
            {
            case '+':
```

```
        {
          n3 = n1 + n2;
          break;
        }
        case '-':
        {
          n3 = n2 - n1;
          break;
        }
        case '*':
        {
          n3 = n1 * n2;
          break;
        }
        case '/':
        {
          n3 = n2 / n1;
          break;
        }
        }
        push(n3);
      }
      e++;
    }
    printf("\nThe result of expression
%s  =  %d\n\n",exp,pop());
    return 0;
```

```
}
```