Page No.
Date

LAB-2

NAME: ANCHUL H. SURANA
USN: 1BM19CS020

1) WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide)

```
#include <stdio.h>
#define MAX 100
char stack [MAX];
int top = -1 ;
void push (char ch)
{
    If ( top = MAX -1)
    printf (" Stack is full \n");
    else
    {   top ++ ;
        stack [top] = ch;
    }
}

char pop()
{   char item;
    if (top == -1)
    printf (" Stack is empty "\n);
    else
    {  item = stack [top] ;
    top -- ;
    return item;
    }
}
```

Anchul H. Surana

NAME: ANCHUL H SURANA
USN: 1BM19CS020

```
int stackempty()
{ if (top==-1)    return 1;
        else    return 0;
}
char stacktop()
{ if (top==-1)
   printf("\n Stack is empty \n");
   else
   return stack[top];
}
int priority (char ch)
{   switch (ch)
    { case '+';
      case '-';   return (1);
      case '*';
      case '/';   return (2);
      case '^';   return (3);
      default : return (0);
    }
}
int main (int argc, char **argv)
{   char infix[100];
      int i, item;
      printf("Enter the infix expression:");
      scanf("%s", infix);
      printf("Expression: %s", infix);
      printf("\n Postfix: ");
      i=0;
      while (infix[i]!='\0')
{   case'(' push (infix[i]);
      break;
```

Anchul H. Surana

NAME: ANKHUL H. SURANA
USN: 1BM19CS020

```
case ')' while ((item = pop()) != '(')
        printf ("%c", item);
        break;
case '+' :
case '-' :
case '*' :
case '/' :
case '^' :
while (!stackempty() && priority(infix[i]);
{       item= pop();
        printf("%c", item);
}
  push (infix[i]);
  break;
default: printf ("%c", infix[i]);
        break;
}
  i++;
}
while (!stackempty() )
{ char item;
    item pop();
    printf("%c", item);
}
printf("\n");
return 0;
}
```

Ankhul H Surana.