Program

```
#include<stdio.h>
#include<string.h>
int prec(char c) {
        if(c=='^')
                return 3;
        else if(c=='/'||c=='*')
                return 2;
        else if(c=='+'||c=='-')
                return 1;
        else
                return -1;
}
char associativity(char c) {
        if(c=='^')
                return 'r';
        return 'l';
}
void infixToPostfix(char s[]) {
        char result[100];
        int k=0;
        int len=strlen(s);
        char stack[100];
        int j=-1;
        for(int i=0; i<len; i++) {
                char c=s[i];
                if((c>='a'&&c<='z')||(c>='A'&&c<='Z')||(c>='0'& c<='9')) {
                        result[k] = c;
                        k++;
                }
                else if(c=='(') {
                        j++;
                        stack[j] = c;
                }
                else if(c==')') {
                        while(j>= 0&&stack[j]!='(') {
                                result[k]=stack[j];
                                k++;
                                j--;
                        }
                        j--;
                }
                else {
                        while(j>=0&&(prec(s[i])<prec(stack[j])||prec(s[i])==prec(stack[j])
&&associativity(s[i])=='l')) {
                                result[k] = stack[j];
                                k++;
                                j--;
                        }
                        j++;
                        stack[j] = c;
                }
        }
        while(j>=0) {
                result[k] = stack[j];
                k++;
                j--;
        }

        result[k]='\0';
        printf("%s\n",result);
```

Experiment No: 6
Date:

INFIX TO POSTFIX CONVERSION

Aim:

To convert a given infix expression to postfix expression and display it.

Algorithm:

1. Start
2. Create function prec(char c)

```
prec(char c)
    if c='^'
        return 3
    else if c='/' or '*'
        return 2
    else if c='+' or '-'
        return 1
    else
        return -1
    end if
end function
```

3. Create function associativity(char c)

```
associativity(char c)
    if c='^'
        return 'r'
    end if
    return 'l'
end function
```

4. Create function InfixToPostfix

```
InfixToPostfix(char s[])
    create 2 character arrays result[100] and stack[100]
    declare int k=0,j=-1 and len=length of string array s
    declare char c
    for (i=0 to len)
        c=s[i]
        if(c is between a and z OR between 0 and 9)
            result[k]=c
            increment k
```

```
}
void main() {
        char exp[100];
        printf("Enter infix expression: ");
        scanf("%s",exp);
        printf("Postfix expression is: ");
        infixToPostfix(exp);
}
```

Output

```
Enter infix expression: 3+2-a/b^6^k*d+1
Postfix expression is: 32+ab6k^^/d*-1+
```

```
            else if (c='(')
                increment j
                stack[j]=c
            else if (c=')')
                while(j>=0 AND stack[j] NOT EQUAL TO '('
                    add element in stack[j] to result[k]
                    increment k and decrement j
                decrement j
            else
                while(j>=0 AND (prec(s[i])<prec(stack[j])) OR
                prec(s[i])=prec(stack[j]) AND associativity(s[i]='l'))
                    add element in stack[j] to result[k]
                    increment k and decrement j
                increment j
                add c to stack[j]
            end if
        end for
        while(j>=0)
            add element in stack[j] to result[k]
            increment k and decrement j
        end while
        result[k]='\0'
        display result array
    end function
```

5. Create Main function

```
    main()
        create character array exp[100]
        read infix expression from the user and store it to exp
        call InfixToPostfix(exp)
    end function
```

6. Stop

Result:

Program has been executed successfully and obtained the output