

Practice Programs

1)WAP to convert a given valid parenthesized infix arithmetic expression to prefix expression. The expression consists of single character operands and the binary operators

+ (plus), - (minus), * (multiply) and / (divide).

CODE:

```
#include <stdio.h>
#include <string.h>
#include <process.h>
int F(char symbol)
{
    switch (symbol)
    {
        case '+':
        case '-':
            return 1;
        case '*':
        case '/':
            return 3;
        case '^':
        case '$':
            return 6;
        case ')':
            return 0;
        case '#':
            return -1;
        default:
            return 8;
    }
}
int G(char symbol)
{
    switch (symbol)
    {
        case '+':
        case '-':
            return 2;
        case '*':
        case '/':
            return 4;
        case '^':
        case '$':
            return 5;
```

```

    case '(':
        return 0;
    case ')':
        return 9;
    default:
        return 7;
    }
}

void infix_prefix(char infix[], char prefix[])
{
    int top, j, i;
    char s[30], symbol;
    top = -1;
    s[++top] = '#';
    j = 0;
    strrev(infix);
    for (i = 0; i < strlen(infix); i++)
    {
        symbol = infix[i];
        while (F(s[top]) > G(symbol))
        {
            prefix[j] = s[top--];
            j++;
        }
        if (F(s[top]) != G(symbol))
        {
            s[++top] = symbol;
        }
        else
        {
            top--;
        }
    }
    while (s[top] != '#')
    {
        prefix[j++] = s[top--];
    }
    prefix[j] = '\0';
    strrev(prefix);
}

void main()
{
    char infix[30], prefix[30];

```

```

    printf("Enter the valid infix expression:\n");
    scanf("%s", infix);
    infix_prefix(infix, prefix);
    printf("The prefix expression is:\n");
    printf("%s\n", prefix);
}

```

OUTPUT:

```

Enter the valid infix expression:
("(a-b/c)*(a/k-1)")
The prefix expression is:
*"-a/bc"-/ak1

Process returned 0 (0x0)   execution time : 4.441 s
Press any key to continue.

```

2) WAP to demonstrate the Evaluation of postfix expression.

CODE:

```

#include<stdio.h>
#include<math.h>
#include<string.h>
double compute(char symbol,double op1,double op2)
{
    switch(symbol)
    {
        case '+':return op1+op2;
        case '-':return op1-op2;
        case '*':return op1*op2;
        case '/':return op1/op2;
        case '$':
        case '^':return pow(op1,op2);
    }
}
void main()
{
    double s[20];
    double res;
    double op1,op2;
    int top,i;
    char postfix[20],symbol;
    printf("Enter the postfix expression:\n");
    scanf("%s",postfix);
    top=-1;
}

```

```

for(i=0;i<strlen(postfix);i++)
{
    symbol=postfix[i];
    if(isdigit(symbol))
        s[++top]=symbol-'0';
    else{
        op2=s[top--];
        op1=s[top--];
        res=compute(symbol,op1,op2);
        s[++top]=res;
    }
}
res=s[top--];
printf("Result= %.2f\n",res);
}

```

OUTPUT:

```

Enter the postfix expression:
53+62/*35*+
Result= 39.00

Process returned 14 (0xE)   execution time : 2.111 s
Press any key to continue.

```

3) WAP to perform factorial of a number using Recursion.

CODE:

```

#include<stdio.h>
long int factorial(int n);
int main() {
    int n;
    printf("Enter a positive integer: ");
    scanf("%d",&n);
    printf("Factorial of %d = %ld", n, factorial(n));
    return 0;
}

long int factorial(int n) {
    if (n>=1)
        return n*factorial(n-1);
    else
        return 1;
}

```

OUTPUT:

```
Enter a positive integer: 5
Factorial of 5 = 120
Process returned 0 (0x0)   execution time : 5.623 s
Press any key to continue.
```

```
Enter a positive integer: 10
Factorial of 10 = 3628800
Process returned 0 (0x0)   execution time : 6.178 s
Press any key to continue.
```

4) WAP to perform GCD of two numbers using Recursion.

CODE:

```
#include <stdio.h>
int hcf(int n1, int n2);
int main() {
    int n1, n2;
    printf("Enter two positive integers: ");
    scanf("%d %d", &n1, &n2);
    printf("G.C.D of %d and %d is %d.", n1, n2, hcf(n1, n2));
    return 0;
}

int hcf(int n1, int n2) {
    if (n2 != 0)
        return hcf(n2, n1 % n2);
    else
        return n1;
}
```

OUTPUT:

```
Enter two positive integers: 366 60
G.C.D of 366 and 60 is 6.
Process returned 0 (0x0)   execution time : 6.656 s
```