**Assignment – 3**

**Q1. Implementation of conversion from infix expression to postfix expression.**

#include <stdio.h>  
#define n 10  
  
char stack[n];  
int top = -1;  
  
void push(char);  
char pop();  
int precedence(char);  
void infix\_to\_postfix(char[], char[]);  
  
int main() {  
 char infix[30], postfix[30];  
 printf("Enter the infix expression: ");  
 scanf("%s", infix);  
 infix\_to\_postfix(infix, postfix);  
 printf("The equivalent postfix expression is %s\n", postfix);  
 return 0;  
}  
  
void push(char ch) {  
 if (top == n) {  
 printf("Stack Overflow\n");  
 } else {  
 stack[++top] = ch;  
 }  
}  
  
char pop() {  
 if (top == -1) {  
 printf("Stack Underflow\n");  
 return -1;  
 }  
 return stack[top--];  
}  
  
int precedence(char ch) {  
 switch (ch) {  
 case '+': case '-':  
 return 1;  
  
 case '\*': case '/':  
 return 2;  
  
 case '^':  
 return 3;  
  
 default:  
 printf("Invalid symbol to return precedence");  
 return -1;  
 }  
}  
  
void infix\_to\_postfix(char infix[], char postfix[]) {  
 int i = 0, j = 0;  
 char ch;  
 push('(');  
  
 while (infix[i]) {  
 switch (infix[i]) {  
 case '(':  
 push(infix[i]);  
 break;  
  
 case ')':  
 ch = pop();  
  
 while (ch != '(') {  
 postfix[j++] = ch;  
 ch = pop();  
 }  
 break;  
  
 case '+': case '-': case '\*': case '/': case '^':  
 while (stack[top] != '(' && precedence(infix[i]) <= precedence(stack[top])) {  
 postfix[j++] = pop();  
 }  
 push(infix[i]);  
 break;  
  
 default:  
 postfix[j++] = infix[i];  
 }  
 i++;  
 }  
 while (stack[top] != '(') {  
 postfix[j++] = pop();  
 }  
 postfix[j] = '\0';  
}

**Output**:

Enter the infix expression: (A+B)\*(C+D)  
The equivalent postfix expression is AB+CD+\*

**Q2. Implementation of evaluation of postfix expression.**

#include <math.h>  
#include <stdio.h>  
#define n 10  
  
int stack[n];  
int top = -1;  
  
void push(int);  
int pop();  
int postfix\_evaluation(char[]);  
  
int main() {  
 char postfix[30];  
 printf("Enter the postfix expression: ");  
 scanf("%s", postfix);  
 printf("The evaluated postfix expression is %d\n", postfix\_evaluation(postfix));  
 return 0;  
}  
  
void push(int ch) {  
 if (top == n) {  
 printf("Stack Overflow\n");  
 } else {  
 stack[++top] = ch;  
 }  
}  
  
int pop() {  
 if (top == -1) {  
 printf("Stack Underflow\n");  
 return -1;  
 }  
 return stack[top--];  
}  
  
int postfix\_evaluation(char postfix[]) {  
 int i = 0, element1, element2;  
  
 while (postfix[i]) {  
 if (postfix[i] >= '0' && postfix[i] <= '9') {  
 push(postfix[i] - '0');  
 } else {  
 element2 = pop();  
 element1 = pop();  
  
 switch (postfix[i]) {  
 case '+':  
 push(element1 + element2);  
 break;  
 case '-':  
 push(element1 - element2);  
 break;  
 case '\*':  
 push(element1 \* element2);  
 break;  
 case '/':  
 push(element1 / element2);  
 break;  
 case '^':  
 push(pow(element1, element2));  
 break;  
 default:  
 printf("Invalid operator\n");  
 return -1;  
 }  
 }  
 i++;  
 }  
 return pop();  
}

**Output**:

Enter the postfix expression: 53\*22+\*  
The evaluated postfix expression is 60