Experiment 15: INFIX TO PREFIX CONVERSION

Code:

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#define SIZE 100
char stack[SIZE];
int top = -1;
void push(char c) {
   stack[++top] = c;
}
char pop() {
   if(top == -1)
      return -1;
   else
      return stack[top--];
}
int precedence(char c) {
   if(c == '^')
      return 3;
   else if(c == '*' || c == '/')
      return 2;
   else if(c == '+' || c == '-')
      return 1;
   else
      return -1;
```

```
}
void reverse(char *exp) {
   int i, j;
   char temp;
   for(i = 0, j = strlen(exp) - 1; i < j; i++, j--) {
      temp = exp[i];
      exp[i] = exp[j];
      exp[j] = temp;
   }
}
void infixToPrefix(char *infix, char *prefix) {
   int i, j = 0;
   char ch;
   reverse(infix);
   for(i = 0; infix[i]; i++) {
      ch = infix[i];
      if(ch == '(')
         infix[i] = ')';
      else if(ch == ')')
         infix[i] = '(';
   }
   for(i = 0; infix[i]; i++) {
      ch = infix[i];
      if(isalnum(ch))
         prefix[j++] = ch;
      else if(ch == '(')
         push(ch);
      else if(ch == ')') {
         while(top != -1 && stack[top] != '(')
            prefix[j++] = pop();
         pop();
```

```
        } else {
            while(top != -1 && precedence(stack[top]) >= precedence(ch))
                prefix[j++] = pop();
            push(ch);
        }
    }
    while(top != -1)
        prefix[j++] = pop();
    prefix[j] = '\0';
    reverse(prefix);
}
int main() {
    char infix[SIZE], prefix[SIZE];
    printf("Enter infix expression: ");
    scanf("%s", infix);
    infixToPrefix(infix, prefix);
    printf("Prefix expression: %s\n", prefix);
    return 0;
}
```

Output:

```
Enter infix expression: (A+B)*(C-D)
Prefix expression: *+AB-CD


=== Code Execution Successful ===
```