14/10/25

Program 2: Infix to postfix.

Q: WAP to convert a given infix arithmetic expression to postfix expression

⇒
```
Infixtopostfix (exp)
{
  create a stack s
  for( i<=0 to length(exp)-1)
  {
    if 'exp[i] is operand
        res = res + exp[i]
    else if exp[i] is operator
        while ( !s.empty() && Has higher pre
                                (s.top(), exp[i])
        {
            res = res+ s.top()
            S. pop()
        }
        S. push(exp[i])
    else if is opening Parenthesis(exp[i])
        S. push(exp[i])
    else if Is closing Parenthesis (exp[i])
    {
        while ( !s. empty && !Is opening
                        Parenthesis(s.top())
        {
            res = res + s. top()
            S. pop()
        }
        S. pop()
    }
}
```
→

```
while (!s.empty())
{
    res = res + s.top()
    s.pop()
}
return res
}
```

Sneha SB
14/10/25

```
#include <stdio.h>
#include <ctype.h>
#define Max 50
char stack[Max];
int top = -1;
void push(char c){
    stack[++top] = c;
}
char pop(){
    return stack[top--];
}
char peek(){
    return stack[top];
}
int pr(char operand){
    if(operand == '^'){
        return(3);
    } else if(operand == '*' || operand == '%'){
        return(2);
    } else if(operand == '+' || operand == '-'){
```

```
        return (1);
    }else {
        return(0);
    }
}
int main(){
    char postfix[50], infix[50], ch, c;
    int i, k=0;
    printf("enter infix exp: ");
    scanf("%s", infix);
    push('#');
    while((ch=infix[i++]) != '\0'){
        if (ch=='('){
            push(ch);
        }else if (isalnum(ch)){
            postfix[k++]=ch;
        }else if (ch==')'){
            while (stack[top] != '('){
                postfix[k++]=pop();
            }
            pop();
        }else{
            while (pr(stack[top])>pr(ch)){
                postfix[k++]=pop();
            }
            push(ch);
```

```
    }
    while (stack[top] != '#'){
        postfix[k++] = pop();
    }
    postfix[k] = '\0';
    printf("\n postfix expression= %s", postfix);
}
```

Output :-

Enter infix expression : a*b+c*d-e

Postfix expression : ab*cd*+e-

O/P Seen

Sneha SB

14/10/25