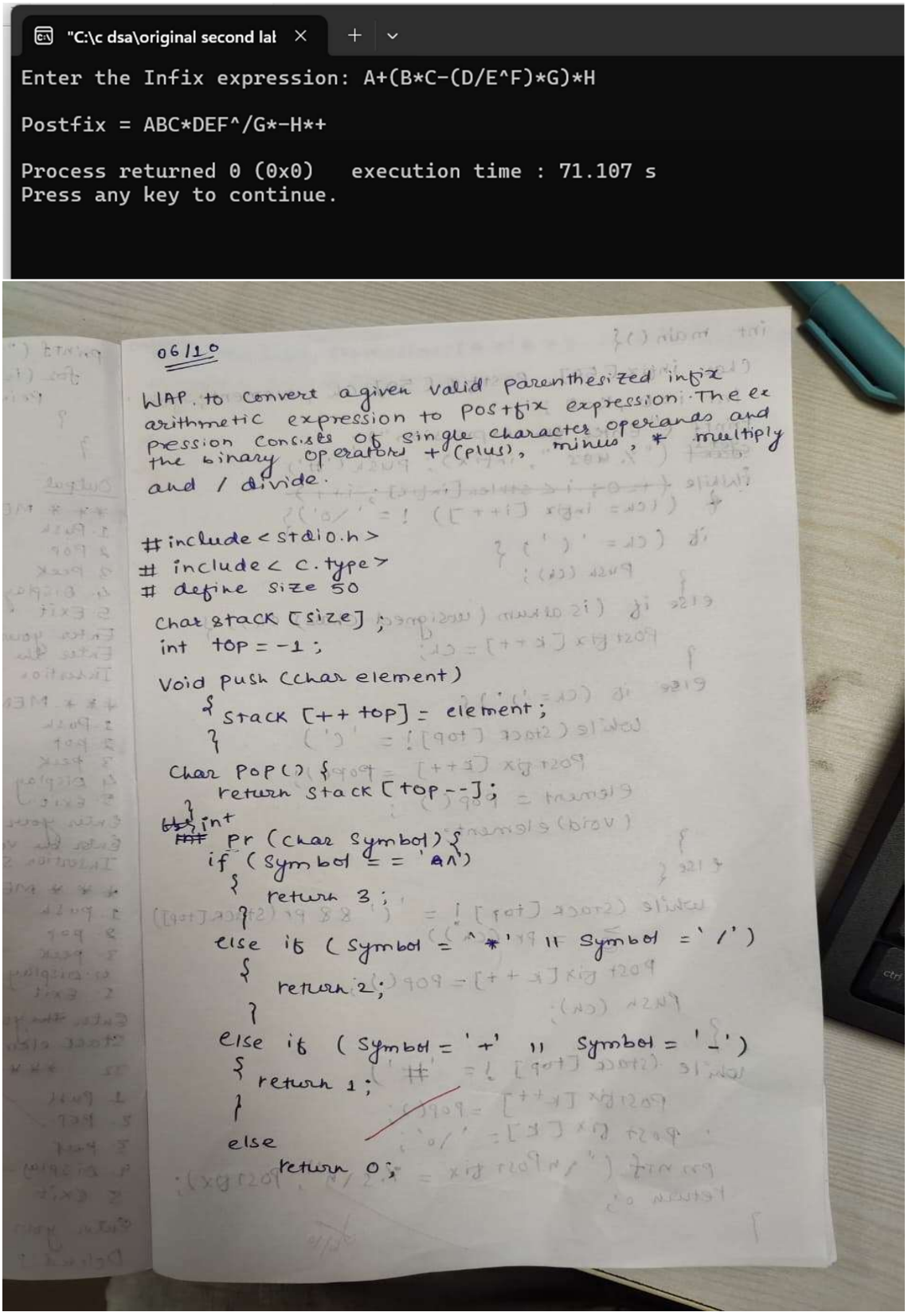
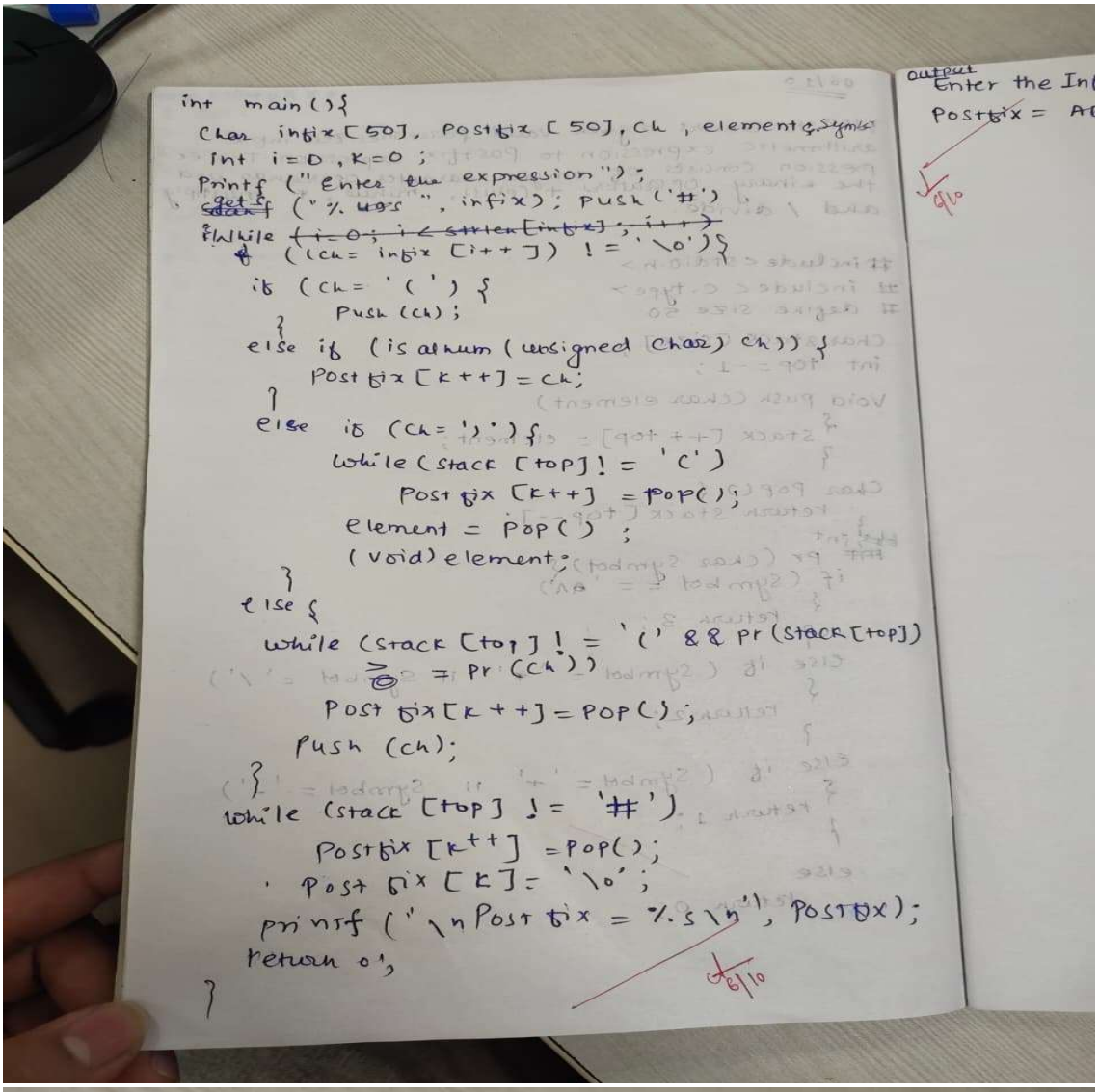
```c
#include <stdio.h>
#include <ctype.h>
#define SIZE 50

char stack[SIZE];
int top = -1;

void push(char element) {
    stack[++top] = element;
}

char pop() {
    return stack[top--];
}

int pr(char symbol) {
    if (symbol == '^') {
        return 3;
    } else if (symbol == '*' || symbol == '/') {
        return 2;
    } else if (symbol == '+' || symbol == '-') {
        return 1;
    } else {
        return 0;
    }
}

int main() {
    char infix[50], postfix[50], ch, element;
    int i = 0, k = 0;
    printf("Enter the Infix expression: ");
    scanf("%49s", infix);

    push('#');
    while ((ch = infix[i++]) != '\0') {
        if (ch == '(') {
            push(ch);
        } else if (isalnum((unsigned char)ch)) {
            postfix[k++] = ch;
        } else if (ch == ')') {
            while (stack[top] != '(')
                postfix[k++] = pop();
            element = pop();
            (void)element;
        } else {
            while (stack[top] != '(' && pr(stack[top]) >= pr(ch))
                postfix[k++] = pop();
            push(ch);
        }
    }

    while (stack[top] != '#')
        postfix[k++] = pop();

    postfix[k] = '\0';
    printf("\nPostfix = %s\n", postfix);
    return 0;
}
```



06/10

WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide).

```
#include<stdio.h>
#include<ctype.h>
#define size 50
char stack[size];
int top = -1;
Void push(char element)
   { stack[++top] = element;
   }
Char pop() {
   return stack[top--];
}
int
pr(char symbol){
  if(symbol == '^')
   {
     return 3;
   }
  else if(symbol = '*' || symbol = '/')
   {
     return 2;
   }
  else if(symbol = '+' || symbol = '-')
   {
     return 1;
   }
  else
     return 0;
```



```
int main(){
 Char infix[50], postfix[50], ch, element;
 int i=0, k=0;
 Printf("Enter the expression");
 get s("%s", infix); push('#');
 while ((ch= infix[i++]) != '\0'){
    if (ch= '(') {
        Push(ch);
    }
    else if (isalnum(unsigned char) ch)) {
        Postfix[k++] = ch;
    }
    else if (ch= ')') {
        while (stack[top]!= '(')
            Postfix[k++] = pop();
        element = Pop();
        (void) element;
    }
   else {
    while (stack[top] != '(' && pr(stack[top])
        >= pr(ch))
        Postfix[k++] = POP();
     Push(ch);
   }
 while (stack[top] != '#')
        Postfix[k++] = pop();
    Post fix[k] = '\0';
   printf("\n Post fix = %.s\n", postfix);
   return 0;
}
```

output
Enter the In
Postfix = A

9/10

9/10

output
Enter the Infix expression: (A+(B*C-(D/E^F)*G)*H)
Postfix = ABC*DEF^/G*-H*+

9/10