NIRAJ KUMAR BT19CS031 QUESTION 3

Implement pass-1 of a two-pass assembler in C/C++.

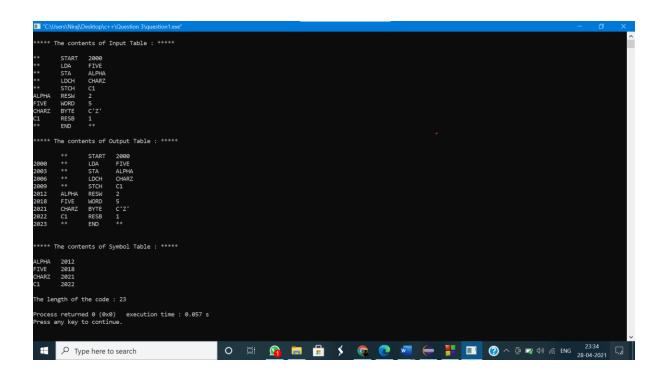
```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#define _GNU_SOURCE
#include <assert.h>
#include <stdlib.h>
#include <stdlib.h>
void passOne(char label[10], char opcode[10], char operand[10], char
code[10], char mnemonic[3]);
void display();
int main()
   // for reading from input
    char label[10], opcode[10], operand[10];
    // for reading from optab
    char code[10], mnemonic[3];
   // call the function
   passOne(label, opcode, operand, code, mnemonic);
   return 0;
}
void passOne(char label[10], char opcode[10], char operand[10], char
code[10], char mnemonic[3])
    int locctr, start, length;
    FILE *fp1, *fp2, *fp3, *fp4, *fp5;
                                       // file pointers
    // read mode
```

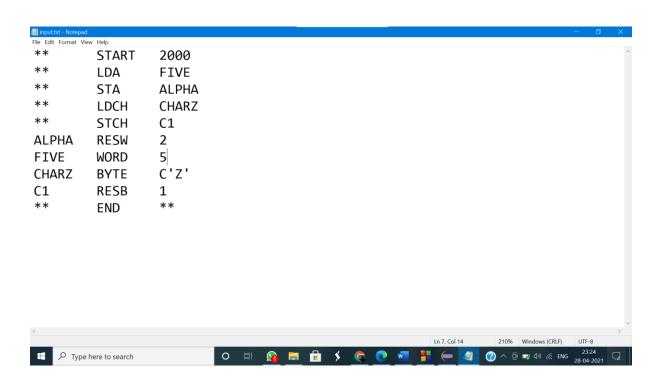
```
fp1 = fopen("input.txt", "r");
   fp2 = fopen("optab.txt", "r");
    // write mode
    fp3 = fopen("symtab.txt", "w+");
    fp4 = fopen("intermediate.txt", "w+");
    fp5 = fopen("length.txt", "w+");
   fscanf(fp1, "%s\t%s\t%s", label, opcode, operand);
                                                                //
read first line
    if (strcmp(opcode, "START") == 0) {
        // atoi() requires stdlib.h header file
        start = atoi(operand);
        // convert operand value from string to integer and assign
to start
        locctr = start;
        fprintf(fp4, "\t%s\t%s\t%s\n", label, opcode, operand);
        // write to output file (additional tab space as start will
not have any locctr)
        fscanf(fp1, "%s\t%s\t%s", label, opcode, operand);
        // read next line
    else {
        locctr = 0;
    // iterate till end
    while (strcmp(opcode, "END") != 0) {
        // 1. transfer address and read line to output file
        fprintf(fp4, "%d\t%s\t%s\t%s\n", locctr, label, opcode,
operand);
        // 2. make symtab file with values not starting with **
        if (strcmp(label, "**") != 0) {
            fprintf(fp3, "%s\t%d\n", Label, locctr);
        }
        // 3. read from optab (code and mnemonic value)
        fscanf(fp2, "%s\t%s", code, mnemonic);
        // 4. traverse till the end of optab file
        while (strcmp(code, "END") != 0) {
            if (strcmp(opcode, code) == 0) {
            // if opcode in input matches the one in optab,
increment locctr by 3
                locctr += 3;
                break;
            }
```

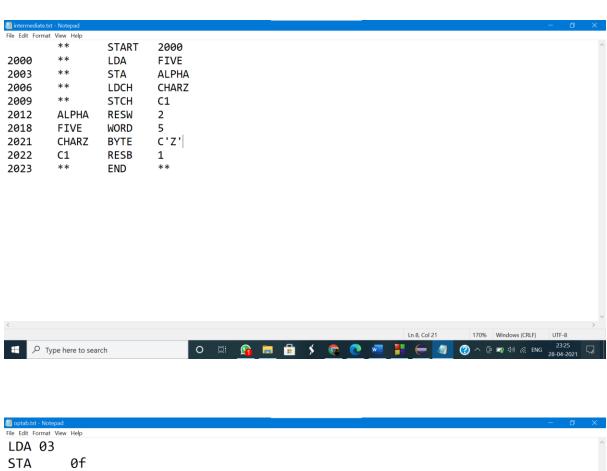
```
// read next line
           fscanf(fp2, "%s\t%s", code, mnemonic);
        }
        // 5. Searching opcode for WORD, RESW, BYTE, RESB keywords
and updating locctr
                // WORD -> add 3 to locctr
        if (strcmp(opcode, "WORD") == 0) {
            locctr += 3;
                // RESW -> add 3*operand to locctr
        else if (strcmp(opcode, "RESW") == 0) {
            locctr += (3 * (atoi(operand)));
             // convert operand to integer and multiply with 3
        }
                // BYTE -> add 1 to locctr
        else if (strcmp(opcode, "BYTE") == 0) {
            ++locctr;
        }
                // RESB -> add operand to locctr
        else if (strcmp(opcode, "RESB") == 0) {
            locctr += atoi(operand);
        }
        // read next line
        fscanf(fp1, "%s\t%s\t%s", label, opcode, operand);
    // 6. transfer last line to file
    fprintf(fp4, "%d\t%s\t%s\t%s\n", locctr, label, opcode,
operand);
    // 7. Close all files
   fclose(fp4);
   fclose(fp3);
   fclose(fp2);
   fclose(fp1);
    // 8. display outputs
    display();
    // 9. calculate length of program
    length = locctr - start;
    fprintf(fp5, "%d", length);
   fclose(fp5);
    printf("\nThe Length of the code : %d\n", length);
}
void display() {
```

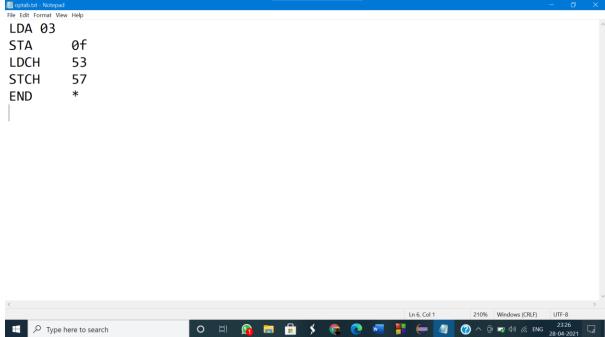
```
char str;
    FILE *fp1, *fp2, *fp3;
    // 1. Input Table
    printf("\n***** The contents of Input Table : *****\n\n");
    fp1 = fopen("input.txt", "r");
    str = fgetc(fp1);
    while (str != EOF) {
        printf("%c", str);
        str = fgetc(fp1);
   fclose(fp1);
   //2. Output Table
   printf("\n\n***** The contents of Output Table : *****\n\n");
   fp2 = fopen("intermediate.txt", "r");
   str = fgetc(fp2);
    while (str != EOF) {
        printf("%c", str);
        str = fgetc(fp2);
   fclose(fp2);
   // 3. Symtable
    printf("\n\n***** The contents of Symbol Table : *****\n\n");
   fp3 = fopen("symtab.txt", "r");
    str = fgetc(fp3);
    while (str != EOF)
        printf("%c", str);
        str = fgetc(fp3);
   fclose(fp3);
}
```

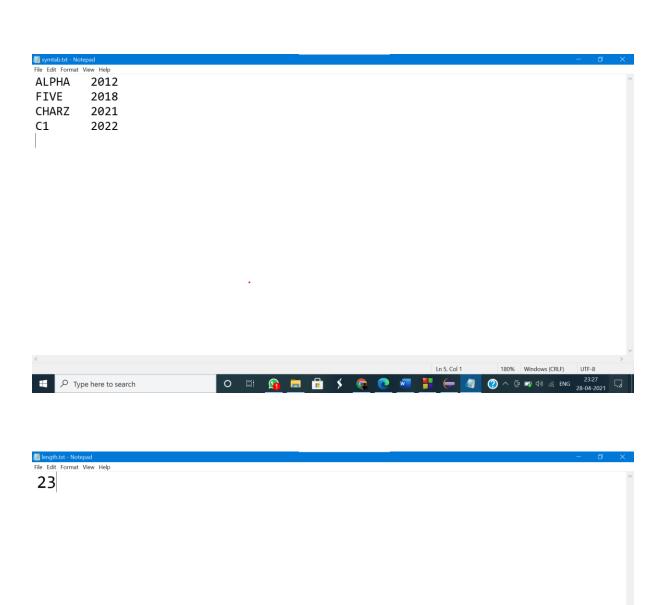
OUTPUT











O 🛱 😘 🛅 🕏 🦠 💿 🚾 👭 🧁 💋 🕖 ^ 🖟 🗖 All (2) ^ All (2) - All

Type here to search