## 7. Implement Pass 1 of a Two-Pass Assembler.

```
#include<stdio.h>
#include<string.h>

void main()
{
    FILE *f1, *f2, *f3, *f4, *flen;
    int lc, sa, op1, o, len; // locctr, starting addr, operand,
machine code, length of byte string
    char m1[20], la[20], op[20], otp[20]; // mnemonic, label, opcode,
opcode in optab
    f1 = fopen("input.txt", "r");
    f3 = fopen("symtab.txt", "w");
    f4 = fopen("out1.txt",  "w");
    fscanf(f1, "%s %s %d", la, m1, &op1);
    if(strcmp(m1, "START")==0) {
        sa = op1;
        lc = sa;
        printf("-\t%s\t%s\t%d\n", la, m1, op1);
        fprintf(f4,  "-\t%s\t%s\t%d\n", la, m1, op1);
    }
    else
        lc = 0;
    fscanf(f1, "%s %s", la, m1);
    while(!feof(f1)) {
        fscanf(f1, "%s", op);
        printf("\n%d\t%s\t%s\t%s\n", lc, la, m1, op);
        fprintf(f4,  "%d\t%s\t%s\t%s\n", lc, la, m1, op);
        if(strcmp(la, "-")!=0)
            fprintf(f3, "\n%d\t%s\n", lc, la);
        f2 = fopen("optab.txt", "r");
        fscanf(f2, "%s %d", otp, &o);
        while(!feof(f2)) { // check if mnemonic opcode is there in
optab
            if(strcmp(m1, otp)==0) {
                lc += 3;
                break;
            }
            fscanf(f2, "%s %d", otp, &o);
        }
        fclose(f2);
        if(strcmp(m1, "WORD")==0)
            lc += 3;
        else if(strcmp(m1, "RESW")==0) {
            op1 = atoi(op);
            lc += (3*op1);
        }
        else if(strcmp(m1, "BYTE")==0) {
            if(op[0]=='X') // hex value
                lc += 1;
            else { // char const
                len = strlen(op)-2;
                lc += len;
            }
        }
        else if(strcmp(m1, "RESB")==0) {
            op1 = atoi(op);
            lc += op1;
        }
        fscanf(f1, "%s%s",  la, m1);
    }
    if(strcmp(m1, "END")==0) {
        printf("Program length: %d\n\n",  lc-sa);
        flen = fopen("length.txt",  "w");
        fprintf(flen,  "%d\n",  lc-sa);
        fclose(flen);
    }
    fclose(f1);
    fclose(f3);
    fclose(f4);
}
```

<u>input.txt</u>

```
copy    START   1000
-       LDA     ALPHA
-       ADD     ONE
-       SUB     TWO
-       STA     BETA
ALPHA   BYTE    C'HOWDY
ONE     RESB    2
TWO     WORD    5
BETA    RESW    1
-       END     -
```

<u>symtab.txt</u>

```
1012     ALPHA

1017     ONE

1019     TWO

1022     BETA
```

<u>out1.txt</u>

```
-  copy  START 1000
1000     -     LDA   ALPHA
1003     -     ADD   ONE
1006     -     SUB   TWO
1009     -     STA   BETA
1012     ALPHA BYTE  C'HOWDY
1017     ONE   RESB  2
1019     TWO   WORD  5
1022     BETA  RESW  1
1025     -     END   -
```

8. **Implement Pass 2 of a Two-Pass Assembler.**

```c
#include<stdio.h>
#include<string.h>
#include<ctype.h>

void main()
{
   FILE *fint, *ftab, *flen, *fsym, *fout;
   int op1[10], txtlen, txtlen1, i, j = 0, len;
   char add[5], symadd[5], op[5], start[10], temp[30], line[20],
label[20], mne[10], operand[10], symtab[10], opmne[10];
   fint = fopen("out1.txt", "r");
   flen = fopen("length.txt", "r");
   ftab = fopen("optab.txt", "r");
   fsym = fopen("symtab.txt", "r");
   fout = fopen("output.txt", "w");
   fscanf(fint, "%s%s%s%s", add, label, mne, operand);
   if(strcmp(mne, "START")==0) {
        strcpy(start, operand);
        fscanf(flen, "%d", &len);
        fclose(flen);
   }
   printf("H^%s^%6s^%06d\nT^00%s^", label, start, len, start);
   fprintf(fout, "H^%s^%s^%d\nT^00%s^", label, start, len, start);
   fscanf(fint, "%s%s%s%s", add, label, mne, operand);
   while(strcmp(mne, "END")!=0) {
        fscanf(ftab, "%s%s", opmne, op);
        while(!feof(ftab)) {
             if(strcmp(mne, opmne)==0) {
                  fclose(ftab);
                  fscanf(fsym, "%s%s", symadd, symtab);
                  while(!feof(fsym)) {
                       if(strcmp(operand, symtab)==0) {
                            printf("%s%s^", op, symadd);
                            fprintf(fout, "%s%s^", op, symadd);
                            break;
                       }
                       else
                            fscanf(fsym, "%s%s", symadd, symtab);
                  }
                  break;
             }
             else
                  fscanf(ftab, "%s%s", opmne, op);
        }
        if((strcmp(mne, "BYTE")==0)||(strcmp(mne, "WORD")==0)) {
             if(strcmp(mne, "WORD")==0) {
                  printf("0000%s^", operand);
                  fprintf(fout, "0000%s^", operand);
             }
             else {
                  len = strlen(operand);
                  for(i = 2;i<len;i++) {
                       printf("%d", operand[i]);
```

```
                        fprintf(fout, "%d", operand[i]);
                    }
                    printf("^");
                    fprintf(fout, "^");
                }
            }
            fscanf(fint, "%s%s%s%s", add, label, mne, operand);
            ftab = fopen("optab.txt", "r");
            fseek(ftab, SEEK_SET, 0);
        }
    printf("\nE^00%s\n\n", start);
    fprintf(fout, "\nE^00%s\n", start);
    fclose(fint);
    fclose(ftab);
    fclose(fsym);
    fclose(fout);
}
```

<u>length.txt</u>

```
25
```

<u>optab.txt</u>

```
LDA     00
STA     23
ADD     01
SUB     05
```

<u>output.txt</u>

```
H^copy^1000^25
T^001000^001012^011017^051019^231022^7279876889^00005^
E^001000
```

## 9. Implement a Single Pass Assembler.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void main()
{
    FILE *f1, *f2, *f3, *f4, *f5;
    int lc, sa, i = 0, j = 0, m[10], pgmlen, len, k, len1, l = 0;
    char name[10], opnd[10], la[10], mne[10], s1[10], mne1[10],
opnd1[10];
    char lcs[10], ms[10];
    char sym[10], symaddr[10], obj1[10], obj2[10], s2[10], q[10],
s3[10];
    f1 = fopen("input.txt", "r");
    f2 = fopen("optab.txt", "r");
    f3 = fopen("symtab.txt", "w+");
    f4 = fopen("symtab1.txt", "w+");
    f5 = fopen("output.txt", "w+");
    fscanf(f1, "%s%s%s", la, mne, opnd);
    if (strcmp(mne, "START") == 0) {
        sa = atoi(opnd);
        strcpy(name, la);
        lc = sa;
    }
    strcpy(s1, "*");
    fscanf(f1, "%s%s%s", la, mne, opnd);
    while (strcmp(mne, "END") != 0) {
        if (strcmp(la, "-") == 0) {
            fscanf(f2, "%s%s", mne1, opnd1);
            while (!feof(f2)) {
                if (strcmp(mne1, mne) == 0) {
                    m[i] = lc + 1;
                    fprintf(f3, "%s\t%s\n", opnd, s1);
                    fprintf(f5, "%s\t0000\n", opnd1);
                    lc = lc + 3;
                    i = i + 1;
                    break;
                }
                else
                    fscanf(f2, "%s%s", mne1, opnd1);
            }
        }

        else {
            fseek(f3, SEEK_SET, 0);
            fscanf(f3, "%s%s", sym, symaddr);
            while (!feof(f3)) {
                if (strcmp(sym, la) == 0) {
                    sprintf(lcs, "%d", lc);
                    fprintf(f4, "%s\t%s\n", la, lcs);
                    sprintf(ms, "%d", m[j]);
                    j = j + 1;
                    fprintf(f5, "%s\t%s\n", ms, lcs);
                i = i + 1;
                break;
            }
            else
                fscanf(f3, "%s%s", sym, symaddr);
        }
        if (strcmp(mne, "RESW") == 0)
            lc = lc + 3 * atoi(opnd);
        else if (strcmp(mne, "BYTE") == 0) {
            strcpy(s2, "-");
            len = strlen(opnd);
            lc = lc + len - 2;
            for (k = 2; k < len; k++) {
                q[l] = opnd[k];
                l = l + 1;
            }
            fprintf(f5, "%s\t%s\n", q, s2);
            break;
        }
        else if (strcmp(mne, "RESB") == 0)
            lc = lc + atoi(opnd);
        else if (strcmp(mne, "WORD") == 0) {
            strcpy(s3, "#");
            lc = lc + 3;
            fprintf(f5, "%s\t%s\n", opnd, s3);
            break;
        }
    }

    fseek(f2, SEEK_SET, 0);
    fscanf(f1, "%s%s%s", la, mne, opnd);
}
fseek(f5, SEEK_SET, 0);
pgmlen = lc - sa;
printf("H^%s^%d^0%x\n", name, sa, pgmlen);
printf("T^");
printf("00%d^0%x", sa, pgmlen);
fscanf(f5, "%s%s", obj1, obj2);
while (!feof(f5)) {
    if (strcmp(obj2, "0000") == 0)
        printf("^%s%s", obj1, obj2);
    else if (strcmp(obj2, "-") == 0) {
        printf("^");
        len1 = strlen(obj1);
        for (k = 0; k < len1; k++)
            printf("%d", obj1[k]);
    }
    else if (strcmp(obj2, "#") == 0) {
        printf("^");
        printf("%s", obj1);
    }
    fscanf(f5, "%s%s", obj1, obj2);
}
fseek(f5, SEEK_SET, 0);
```

```
    fscanf(f5, "%s%s", obj1, obj2);
    while (!feof(f5)) {
        if (strcmp(obj2, "0000") != 0) {
            if (strcmp(obj2, "-") != 0) {
                if (strcmp(obj2, "#") != 0) {
                    printf("\n");
                    printf("T^%s^02^%s", obj1, obj2);
                }
            }
        }
        fscanf(f5, "%s%s", obj1, obj2);
    }
    printf("\nE^00%d\n", sa);
}
```

<u>input.txt</u>

```
COPY     START      1000
-        LDA        ALPHA
-        STA        BETA
ALPHA    RESW       1
BETA     RESW       1
-        END        -
```

<u>optab.txt</u>

```
LDA      00
STA      23
LDCH     15
STCH     18
```

<u>symtab.txt</u>

```
ALPHA    *
BETA     *
```

<u>symtab1.txt</u>

```
ALPHA    1006
BETA     1009
```

<u>output.txt</u>

```
00       0000
23       0000
1001     1006
1004     1009
```

## 10. Implement a Two-Pass Macro Processor.

```
// PASS 1
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void main()
{
    FILE *f1, *f2, *f3;
    char mne[20], opnd[20], la[20];
    f1 = fopen("inp.txt", "r");
    f2 = fopen("namtab.txt", "w+");
    f3 = fopen("argtab.txt", "w+");
    fscanf(f1, "%s%s%s", la, mne, opnd);
    while (strcmp(mne, "MEND") != 0) {
        if (strcmp(mne, "MACRO") == 0) {
            fprintf(f2, "%s\n", la);
            fprintf(f3, "%s\t%s\n", la, opnd);
        }
        else
            fprintf(f3, "%s\t%s\n", mne, opnd);
        fscanf(f1, "%s%s%s", la, mne, opnd);
    }
    fprintf(f3, "%s", mne);
    fclose(f1);
    fclose(f2);
    fclose(f3);
    printf("Pass 1 is completed\n");
}

// PASS 2
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
void main()
{
    FILE *f1, *f2, *f3, *f4, *f5;
    int i, len;
    char mne[20], opnd[20], la[20], name[20], mne1[20], opnd1[20],
arg[20];
    f1 = fopen("inp.txt", "r");
    f2 = fopen("namtab.txt", "r");
    f3 = fopen("argtab.txt", "r");
    f4 = fopen("atab2.txt", "w+");
    f5 = fopen("op2.txt", "w");
    fscanf(f1, "%s%s%s", la, mne, opnd);
    while (strcmp(mne, "END") != 0) {
        if (strcmp(mne, "MACRO") == 0) {
            fscanf(f1, "%s%s%s", la, mne, opnd);
            while (strcmp(mne, "MEND") != 0)
                fscanf(f1, "%s%s%s", la, mne, opnd);
        }
        else {
            fscanf(f2, "%s", name);
            if (strcmp(mne, name) == 0) {
                len = strlen(opnd);
                for (i = 0; i < len; i++) {
                    if (opnd[i] != ',')
                        fprintf(f4, "%c", opnd[i]);
                    else
                        fprintf(f4, "\n");
                }
                fseek(f2, SEEK_SET, 0);
                fseek(f4, SEEK_SET, 0);
                fscanf(f3, "%s%s", mne1, opnd1);
                fprintf(f5, ".\t%s\t%s\n", mne1, opnd);
                fscanf(f3, "%s%s", mne1, opnd1);
                while (strcmp(mne1, "MEND") != 0) {
                    if ((opnd1[0] == '&')) {
                        fscanf(f4, "%s", arg);
                        fprintf(f5, "-\t%s\t%s\n", mne1, arg);
                    }
                    else
                        fprintf(f5, "-\t%s\t%s\n", mne1, opnd1);
                    fscanf(f3, "%s%s", mne1, opnd1);
                }
            }
            else
                fprintf(f5, "%s\t%s\t%s\n", la, mne, opnd);
        }
        fscanf(f1, "%s%s%s", la, mne, opnd);
    }
    fprintf(f5, "%s\t%s\t%s\n", la, mne, opnd);
    fclose(f1);
    fclose(f2);
    fclose(f3);
    fclose(f4);
    fclose(f5);
    printf("Pass 2 completed\n");
}
```

**Pass 1:**

inp.txt

```
EX1      MACRO &A,&B
-        LDA   &A
-        STA   &B
-        MEND  -
SAMPLE   START 1000
-        EX1   N1,N2
N1       RESW  1
N2       RESW  1
-        END   -
```

namtab.txt

```
EX1
```

argtab.txt
```
EX1 &A,&B
LDA &A
STA &B
MEND
```

**Pass 2:**

atab2.txt
```
N1
N2
```

op2.txt
```
SAMPLE   START 1000
.        EX1   N1,N2
-        LDA   N1
-        STA   N2
N1       RESW  1
N2       RESW  1
-        END   -
```

## 11. Implement an Absolute Loader.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void main()
{
    FILE *fp;
    int addr, staddri;
    char line[50], staddr[10];
    fp = fopen("object_code.txt", "r");
    fscanf(fp, "%s", line);
    while (!feof(fp)) {
        fscanf(fp, "%s", line);
        if (line[0] == 'T') {
            int i = 0, j = 0;
            for (i = 2, j = 0; i < 8; i++, j++)
                staddr[j] = line[i];
            staddr[j] = '\0';
            staddri = atoi(staddr);
            i = 12;
            while (line[i] != '$') {
                if (line[i] != '^') {
                    printf("00%d %c%c\n", staddri, line[i], line[i +
1]);
                    staddri++;
                    i += 2;
                }
                else
                    i++;
            }
        }
        else if (line[0] == 'E')
            break;
    }
}
```

<u>object_code.txt</u>

```
H^SAMPLE^001000^0035
T^001000^0C^001003^071009$
T^002000^03^111111$
E^001000
```

<u>**Output:**</u>

```
001000 00
001001 10
001002 03
001003 07
001004 10
001005 09
002000 11
002001 11
002002 11
```

12. **Implement a Symbol Table with Suitable Hashing.**
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define LENGTH 7

struct hashTable {
    char label[10];
    int addr;
} ht[LENGTH];

void addLabel()
{
    int addr;
    char label[10];
    printf("Enter label name: ");
    scanf("%s", label);
    printf("Enter label address: ");
    scanf("%d", &addr);
    int loc = addr % LENGTH;
    if (ht[loc].addr == -1)
    {
        ht[loc].addr = addr;
        strcpy(ht[loc].label, label);
    }
    else
        printf("Hashtable slot occupied\n");
}

void display()
{
    for (int i = 0; i < LENGTH; i++)
        if (ht[i].addr != -1)
            printf("%d %s\n", ht[i].addr, ht[i].label);
        else
            printf("0 0\n");
}

void search()
{
    char label[10];
    int i, set=0, s;
    printf("Enter label name: ");
    scanf("%s", label);
    for (i=0; i<LENGTH; i++) {
        if (ht[i].addr) {
            if (!strcmp(ht[i].label, la)) {
                set=1;
                s = ht[i].addr;
            }
        }
    }
    if (set)
```

```
        printf("Label is present!\n");
    else printf("Label is not present!\n");
}

void main()
{
    for (int i = 0; i < LENGTH; i++) {
        ht[i].addr = -1;
        strcpy(ht[i].label, "");
    }
    int c = 0;
    while (c < 3) {
        printf("1. Add label. \2. View hashtable. \nENTER CHOICE: ");
        scanf("%d", &c);
        switch (c) {
            case 1:
                addLabel();
                break;
            case 2:
                display();
                break;
            default: exit(0);
        }
    }
}
```

**Output:**

```
1. Add label.
2. View hashtable.
3. Search for label.
ENTER CHOICE: 1
Enter label name: loop
Enter label address: 1275

1. Add label.
2. View hashtable.
3. Search for label.
ENTER CHOICE: 1
Enter label name: clear
Enter label address: 6475

1. Add label.
2. View hashtable.
3. Search for label.
ENTER CHOICE: 1
Enter label name: rdlp
Enter label address: 2467

1. Add label.
2. View hashtable.
3. Search for label.
ENTER CHOICE: 2
6475 clear
1275 loop
```

```
0 0
2467 rdlp
0 0
0 0
0 0

1. Add label.
2. View hashtable.
3. Search for label.
ENTER CHOICE: 3
Enter label name: loop
Label is present!

1. Add label.
2. View hashtable.
3. Search for label.
ENTER CHOICE: 4
```