**C program for implementing an Absolute Loader**

**Absolute Loader?**

An **Absolute Loader** is a simple type of loader used in operating systems to load an object program into memory without modification. It reads the object code and directly places it in memory at the specified address.

**1. Header and Variable Declarations**

```
#include<stdio.h>

#include<string.h>

#include<stdlib.h>

void main()

{

   FILE *fp;

   int i, addrl, l, j, staddrl;

   char name[10], line[50], namel[10], addr[10], rec[10], ch, staddr[10];
```

- Declares variables for reading lines, tracking addresses, etc.
- fp is a file pointer.
- line is used to store a line from the file.
- namel stores the program name from the object file.
- staddr and staddrl store the starting address from the object program.

**2. Input and File Opening**

```
printf("enter program name:");

scanf("%s", name);

fp = fopen("abssrc.txt", "r");
```

- The user is prompted to enter the program name (expected to match the name in the object file).
- Opens the file "abssrc.txt" which contains the object code.

**3. Extract Program Name from Object File**

```
fscanf(fp, "%s", line);

for(i=2, j=0; i<8 && j<6; i++, j++)

   namel[j] = line[i];

namel[i] = '\0';

printf("name from obj.%s\n", namel);
```

- Reads the first line from the file.
- Extracts characters from index 2 to 7 and stores them as namel.
- This assumes the first line is of the form: H^PROG1^001000^00107A
    - The name "PROG1" starts from index 2.
- Then it prints the extracted name.

**4. Check if User-entered Name Matches File Name**

```
if(strcmp(name, namel) == 0)
{
```

- Compares the input name and the one in the object file.

**5. Read Remaining Lines (T-records) and Print Memory Map**

```
do
{
   fscanf(fp, "%s", line);
   if(line[0] == 'T')
   {
      for(i=2, j=0; i<8 && j<6; i++, j++)
         staddr[j] = line[i];
      staddr[j] = '\0';
      staddrl = atoi(staddr);
      i = 12;
      while(line[i] != '$')
      {
         if(line[i] != '^')
         {
            printf("00%d\t%c%c\n", staddrl, line[i], line[i+1]);
            staddrl++;
            i = i + 2;
         }
         else
            i++;
      }
```

```
    }
    else if(line[0] == 'E')
        fclose(fp);
}
while(!feof(fp));
```

**Breakdown:**

- For each line:
    - If it starts with 'T' (text record), it parses the start address.
    - Converts it from string to integer.
    - Then it loops through the rest of the line to extract and print byte pairs (representing machine code) along with the address.
- Increments address for each byte pair.
- Stops when it encounters $ or end-of-file.

**Sample Output Explanation**

**Suppose the abssrc.txt contains:**

H^PROG1^001000^00107A

T^001000^1E^141033^281030^301015^...

E^001000

If the user enters: PROG1, the output will be something like:

name from obj.PROG1

001000  14

001001  10

001002  33

001003  28

001004  10

...

Each line displays:

- Address (in form 00xxxx)
- Corresponding object code bytes