```
void Create_Table(Table& ES_Table, ifstream& inputs, int Total_Length)
{
    char buffer[80];
    inputs.getline(buffer, 80);
    if (buffer[0] != '*')
    {
    cerr << "Incorrect input file format\n";
    exit(1);
    }
    inputs.getline(buffer, 80);
    while (buffer[0] != '*')
    {
    char* symbol;
    int address;
    symbol = strtok(buffer, " ");
    if (ES_Table.Is_In_Table(symbol))
    {
        cerr << "Multiple symbol definition: \"" << symbol << "\"\n";
        exit(1);
    }
    address = atoi(strtok(NULL, "."));
    ES_Table.Put_In_Table(symbol, (address+Total_Length));
    inputs.getline(buffer,80);
    }
}

int Loader_One(Table& ES_Table, ifstream*& inputs, int number_of_files, int&
    Begin_Execution)
{
    ofstream middle;
    middle.open("intermediate");
    int File_Number = 0;
    int Total_Length = 0;
    int PLA = 0;
    while (File_Number < (number_of_files-3))
    {
    Create_Table(ES_Table, inputs[File_Number], Total_Length);
    char buffer[80];
    char* token;
    inputs[File_Number].getline(buffer, 80);
    token = strtok(buffer, "."); //get rid of the H
    if (strcmp(token, "H") != 0)
    {
        cerr << "Incorrect Header Record";
        exit(1);
    }

    token = strtok(NULL, "."); //initial program execution
                    //address
    if (File_Number == 0)
    {
        Begin_Execution = atoi(token);
    }

    token = strtok(NULL, "."); //segment name
    if (ES_Table.Is_In_Table(token))
    {
        cerr << "Segment name \"" << token << "\" multiply defined\n";
        exit(1);
    }
    else
    {
        ES_Table.Put_In_Table(token, Total_Length); //segment name is
    }                                   //an external symbol
    token = strtok(NULL, "."); //PLA
    int temp_PLA = atoi(token);

    token = strtok(NULL, "."); //segment length
```

```
    int Segment_Length = atoi(token);
    Total_Length += Segment_Length;

    token = strtok(NULL, "."); //M?
    if (token == NULL)
    {
        //middle << temp_PLA << '\n';
        cerr << "Attempt to access absolute memory address\n";
        exit(1);
    }
    else
    {
        PLA += temp_PLA;
        middle << PLA << '\n';
        PLA += Segment_Length;
    }

    File_Number++;
    }
    //ES_Table.Put_Table(middle);
    return  Total_Length;
}
```