Project Report

On

**DEVELOPMENT OF MERRIT LIST SOFTWARE IN C**

By

ABHIJEET SANJIV BONDE

(E. No. 19103002)

Jaypee Institute of Information Technology, Noida

**ABSTRACT:**

On this planet, every academic institute needs a robust software which, based on some variable parameters can display the merit list based on marks the students have obtained. So, In this project an application is developed using C language which reads the source information from a text file and calculates the total marks and their percentage and then based on the Users choice, prepare the merit list and displays the same as well as can store the merit list into a text file.

In this project I have used the concepts of C language which majorly deal with Linked List, Strings, Numerical data and Sorting process. File I/O are also used in certain operation in order to get the best results and for the program to be user friendly.

**Existing System:**

Many more existing systems use the manual and are hand run processes to prepare the merit list and declare the result. The User has to manually enter the Name, Enrollment number, and Marks in each subject, then the software is ready to calculate the percentage and merit list according to the marks. This process is very slow to give the result. Such kind of software’s are easily prone to making errors. When a parameter has to be added to such software, it has to be redesigned, adding more efforts. The more the students, the more tedious the work is.

**Proposed System:**

To overcome the Problems that the Existing systems face, this program is designed to take the input from a user defined text file and, the automated generated merit list is stored in a text file which makes the application versatile to use for the user. This makes the user easy to use the application. The program also keeps the record of number of students. Based on the data that has been entered in the text file, the application sorts the data and makes a Merit List. Since the User has to enter the data only once, it makes the application less prone to errors. When the Merit List is prepared, the User can Display it or store it in a File. When a parameter or a record has to be added to the system, minimal changes are required. Number of students in not a issue for this system. The system can be easily scaled according to the needs.

**Advantages:**

1. Very fast, accurate, and robust.
2. No need of any extra manual effort.
3. With just minimal information user can operate the application.
4. Easy to operate
5. Options available for sorting the data on different parameters.
6. A User can create the merit list for all subjects separately, once the marks are entered in the file.

**REQUIREMENTS:**

1. The User has to input the data into a text file.
2. Data consists of Serial Number, Name of the student (First name, Middle name, and the Last name), Marks of 5 subjects (marks 1, marks 2, marks 3, marks 4, marks 5) with spaces in between each field.
3. Run the executable file.
4. Choose an option from the menu displayed.
5. If the user wants to store the Merit list, an option is available.
6. After the User has finished, Memory allocated to the linked list has to be freed to avoid any future errors.

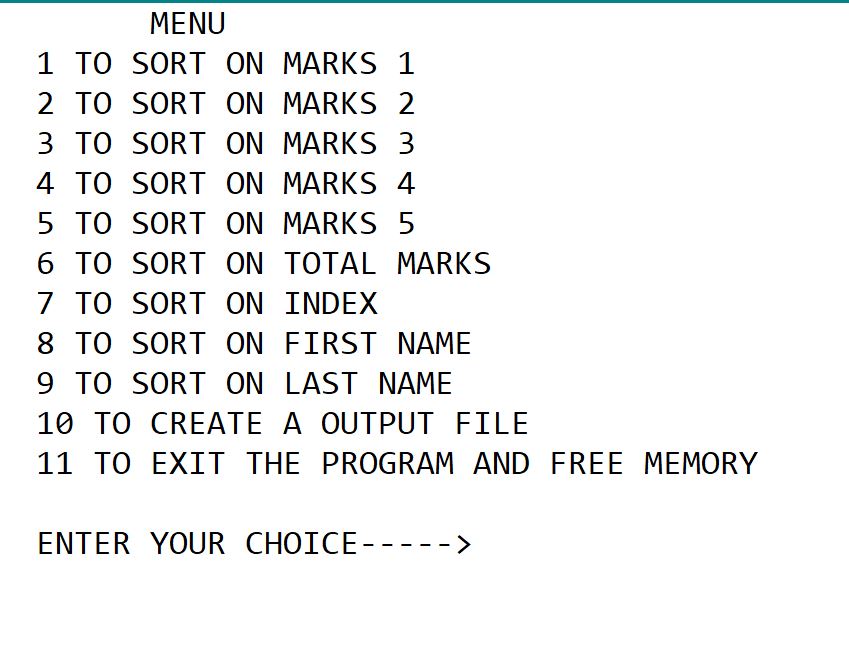
**MODULE DESCRIPTION:**

As, with a well-organized display and system, it is better to navigate through the application. So, the application has been broken down in some simple functions.

1. MENU
2. CREATE LINKED LIST
3. SORTING ON VARIOUS PARAMETERS
4. DISPLAY
5. CREATE OUTPUT FILE
6. FREE MEMORY AND EXIT

**MENU:**

There are a lot of applications out there with just one UI. In this application also there is just one User Interface which is known as menu. It is totally based on C language. It has the following options.



**CREATE LINKED LIST:**

In this module we read the data (i.e. S.No, Name of the student (i.e. First name and Last name) and marks of 5 subjects) from a (.txt) file.

The text file has the following format.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | First name | Middle name | Last name | Marks 1 | Marks 2 | Marks 3 | Marks 4 | Marks 5 |
| 1 | Abhijeet | Sanjiv | Bonde | 21 | 23 | 34 | 56 | 87 |
| 2 | Damon | Joseph | Salvatore | 43 | 56 | 76 | 89 | 09 |
| 3 | Stefan | Joseph | Salvatore | 27 | 54 | 56 | 52 | 56 |
| **:** | **:** | **:** | **:** | **:** | **:** | **:** | **:** | **:** |
| 99 | Elena | David | Gilbert | 85 | 59 | 32 | 58 | 65 |
| 100 | Hemangi | Kunal | Sharma | 34 | 65 | 82 | 45 | 98 |

For testing purpose, 100 students record have been processed but as such there is not limit on the number of students. As input data is taken through a file the same software can be used for generated the merit list of different classes.

The software uses linked list for efficient implementation. Therefore, a structure is required to implement the linked list containing the record of each student. The structure is declared as follows.

|  |
| --- |
| struct list  {  int index;  char f\_name[20];  char m\_name[20];  char l\_name[20];  int m1;  int m2;  int m3;  int m4;  int m5;  int total\_marks;  float percentage;  struct list \*next;  }; |

At the start of this module, the module asks the user for the file name of the input file (as shown in Fig 1.1). If the program is not able to find a file with the given name it will show an error (as shown in Fig 1.2). So, the input file is user desired. The text file is then opened using FILE I/O. Once the file is opened successfully, data from the file is read and then stored in the structure which has been shown above. Two additional fields, total marks and percentage, are computed and added to the data while placing the records in the structure. Then the nodes of the structure are linked and thus the linked list is created. Once the linked list has been created then the file is closed for further simplification. Now all the student’s records are present in the linked list and the same is referred for further processing.

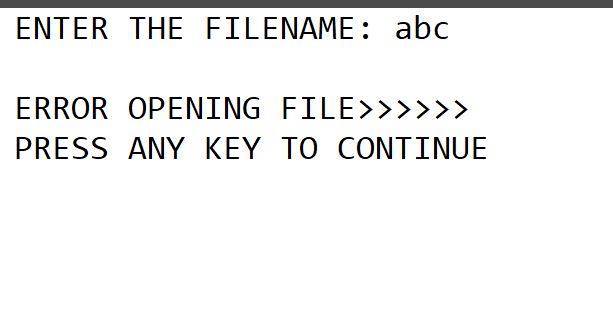
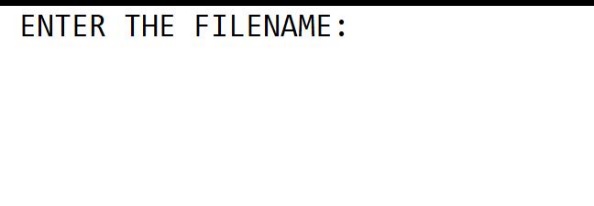
 

Fig 1.1 Fig 1.2

**SORTING:**

In this process the created linked list is sorted accrding to the the users input. User has options whether to sort on marks 1, marks 2, marks 3, marks 4, marks 5, total marks, First name, Last name, and index. In this process the linked list is not sorted but the index will be rearranged. Sorting in here is of 2 types:

1. Sorting based on Integer
2. Sorting based on String

The process of sorting based on Integer is as follows.

Let us assume that the user has selected the option to sort on marks 1. For the sorting to be done two arrays have been created (ie. Index and marks). The sorting is done by comparing the elements in the marks array. From the linked list that has been created before, the index and marks data has been copied in these arrays, respectively. From the field m1 of the linked list the data has been copied in the marks array. Once all the marks are in the marks array, the number of elements in the marks array are counted and then the sorting process starts. In this sorting process Bubble sort has been used to sort the index array. The sorting code is as follows.

|  |
| --- |
| for(i=0; i<(no\_of\_elements-1); i++)  {  for(j=0; j<(no\_of\_elements-i-1); j++)  {  if(marks[j]<marks[j+1])  {  temp=marks[j];  marks[j]=marks[j+1];  marks[j+1]=temp;  temp=index[j];  index[j]=index[j+1];  index[j+1]=temp;  }  }  } |

The sorting is done by comparing the (j)th and (j+1)th elements from the marks array. If the (j)th element is smaller than (j+1)th element then (j)th and (j+1)th elements are interchanged and simultaneously the (j)th and (j+1)th elements of index array are also interchanged, as it will be required for preparing the List (Note that no change has been made in the linked list. The linked list remains the same as it was when created). The new modified index which contains the index values as per the sorted merit list is then passed to the display function and to the Create output file function for the further process.

The process of sorting on string is as follows:

At first, a loop counts the number of elements in the list. A dummy index is created with its each element being (i+1) value, where i is the current position in the index array. Once the index array is filled, the strings (i.e. The first name or the last name) are compared with the help of ***strcmp()*** function (as the function returns, a negative value if the first string precedes the second string alphabetically, a value of zero if the first string and the second string are identical disregarding the case, a positive value if the second string precedes the first string alphabetically). If the output of the ***strcmp()*** function is positive then the elements are interchanged and simultaneously are the index values changed. At last the index is returned from the sort routine.

**DISPLAY:**

In this module the list is printed on the screen as per the index passed to it. In our case the index passed is sorted in ascending order. There are 2 parameters passed to this module (i.e. starting address of the Linked list and the index). With the index number that comes first, the pointer traverses through the linked list to print the record with the associated index. After printing each record, the pointer is again initialized to starting position of the linked list.

**CREATE OUTPUT FILE:**

In this module, the starting address of the linked list and index are the input parameters. It will ask the user to type the name of the file that the user wants to create for storing the result. Then the number of elements are counted according to which the data will be written in the file. The data is written in a systematic manner, as there are proper functions used for the data writing process. The data is written in tabular form using the formatted I/O functions. The code segment which writes the data is as shown in Fig. 1.3.

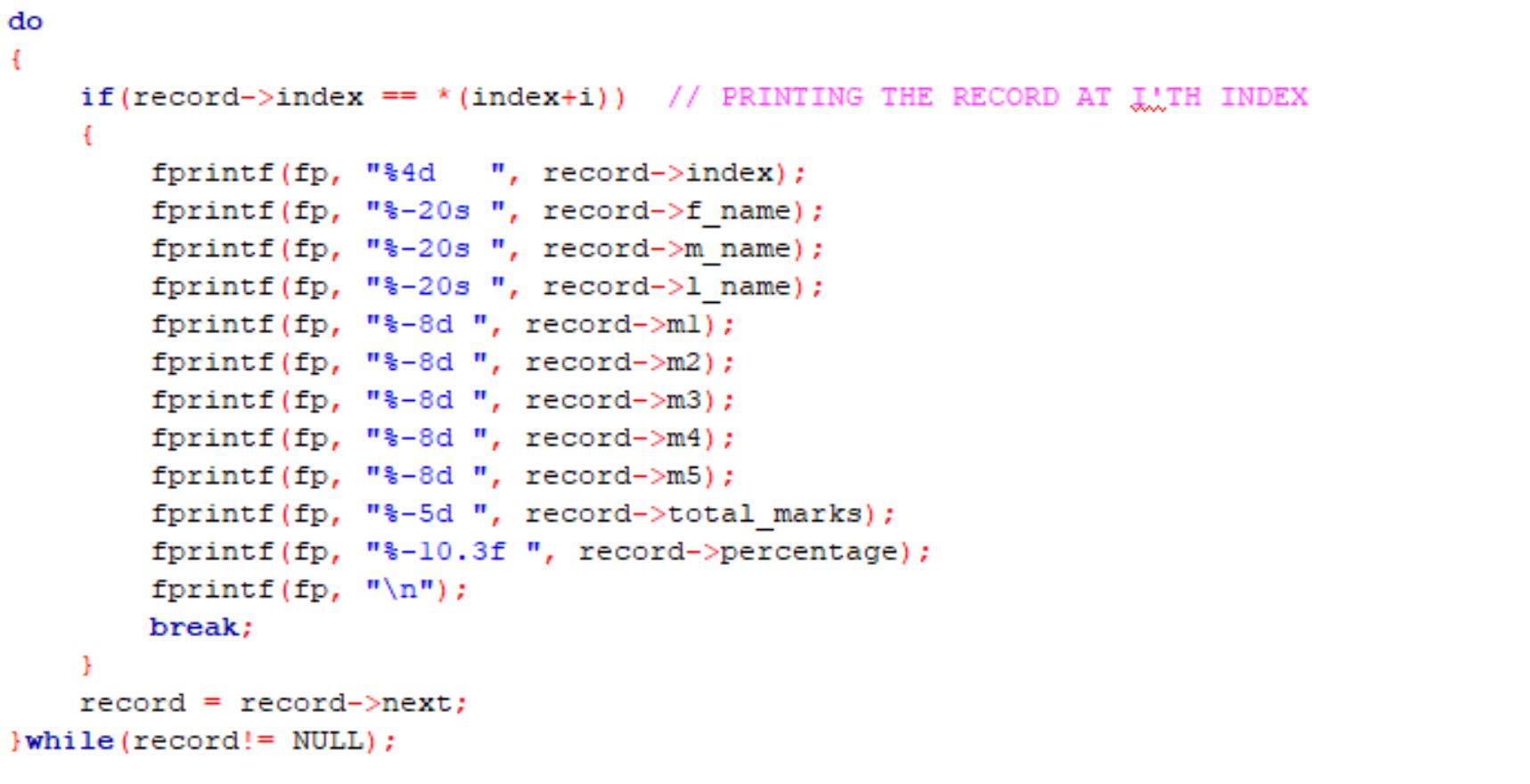


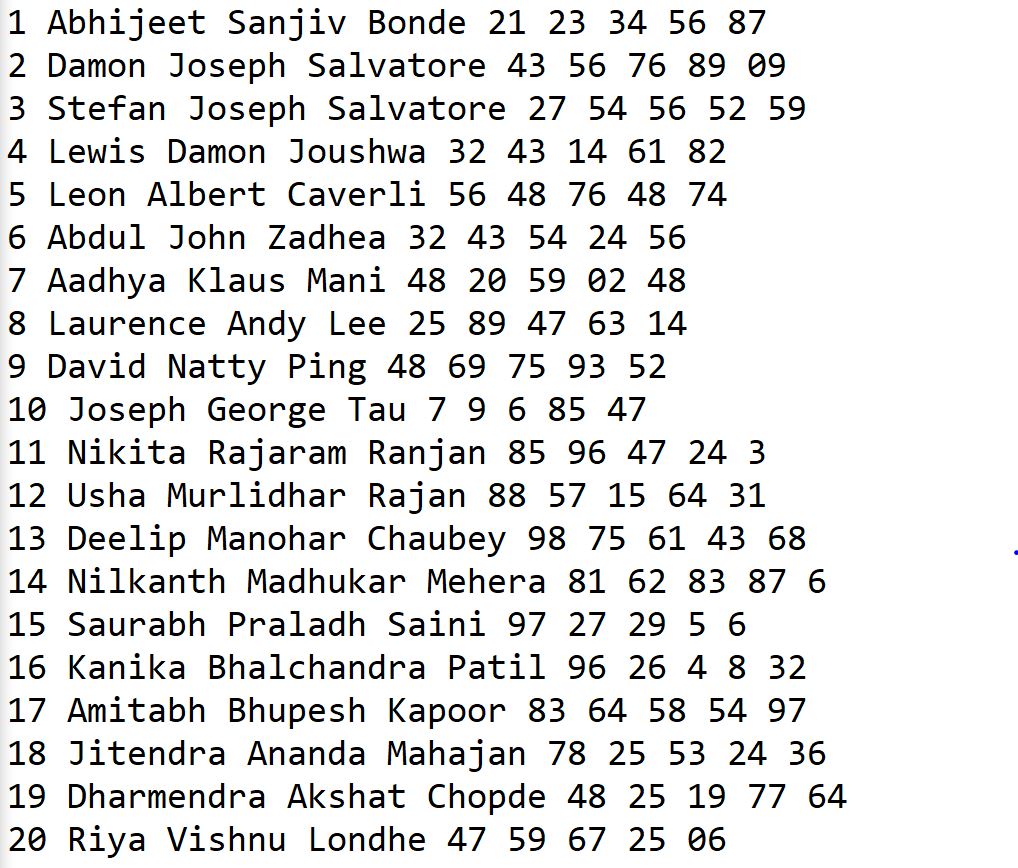
Fig. 1.3

**FREE MEMORY AND EXIT:**

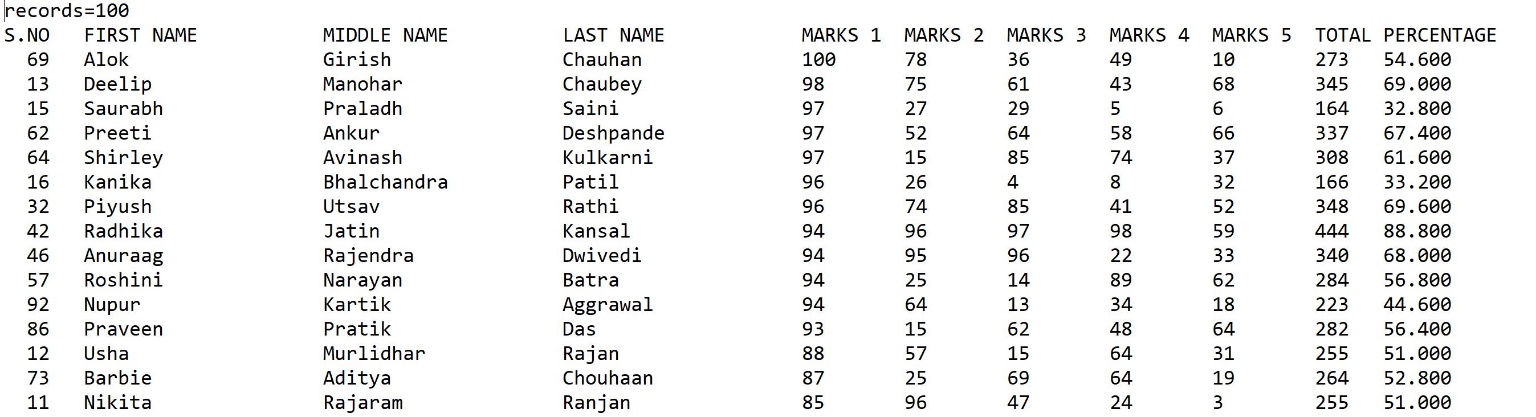
This module is used to deallocate the memory that was reserved for storing the linked list. Once a node is freed the module prints a message onto the screen saying “FREED NODE NO: x” (where x is the node number). At last when the list pointer is pointing at null the program will end and will lead its way to exit.

**COMPARISION BETWEEN INPUT AND OUTPUT:**

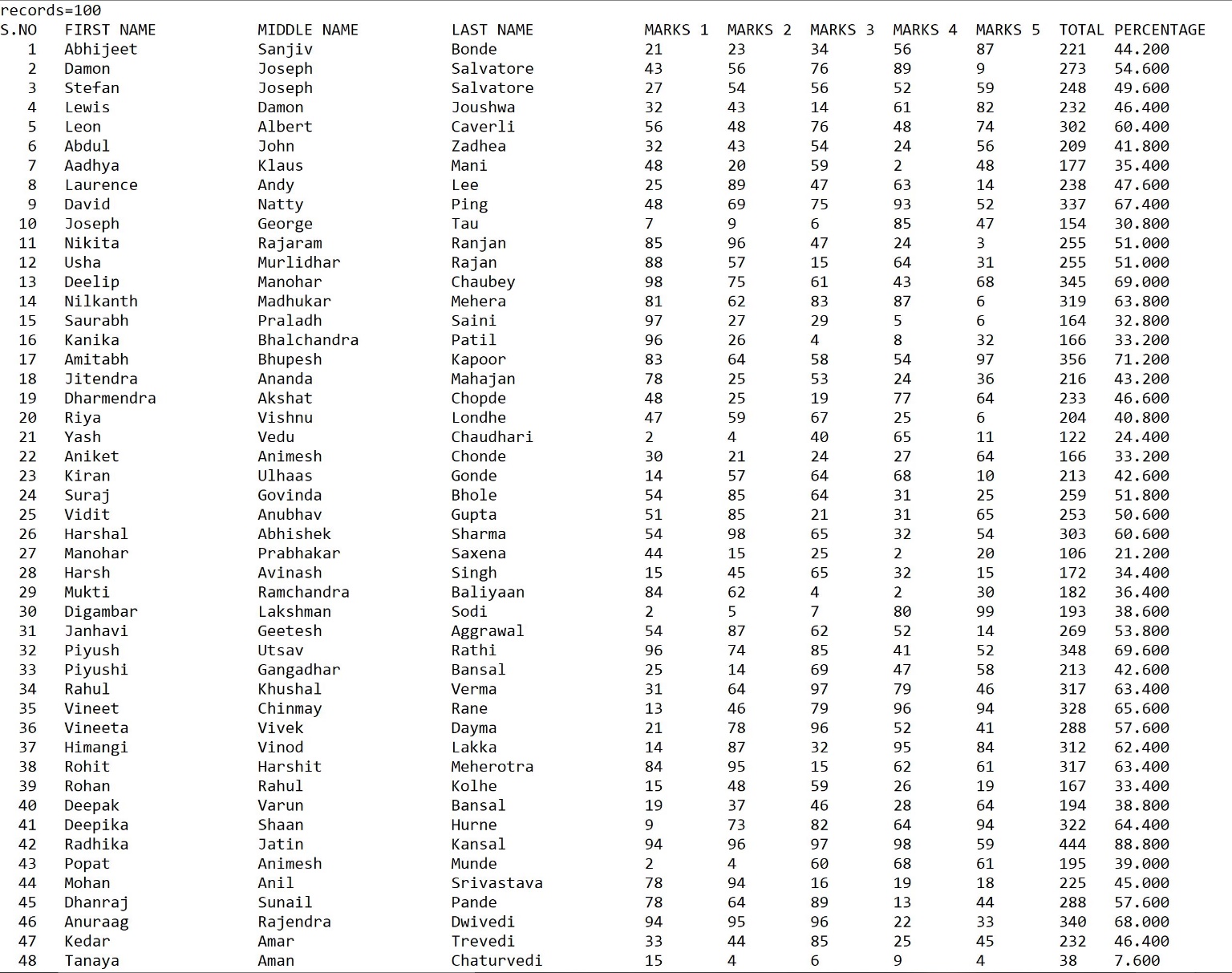
* INPUT:

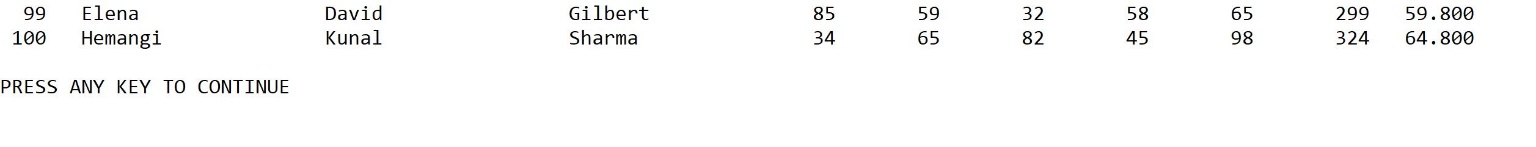
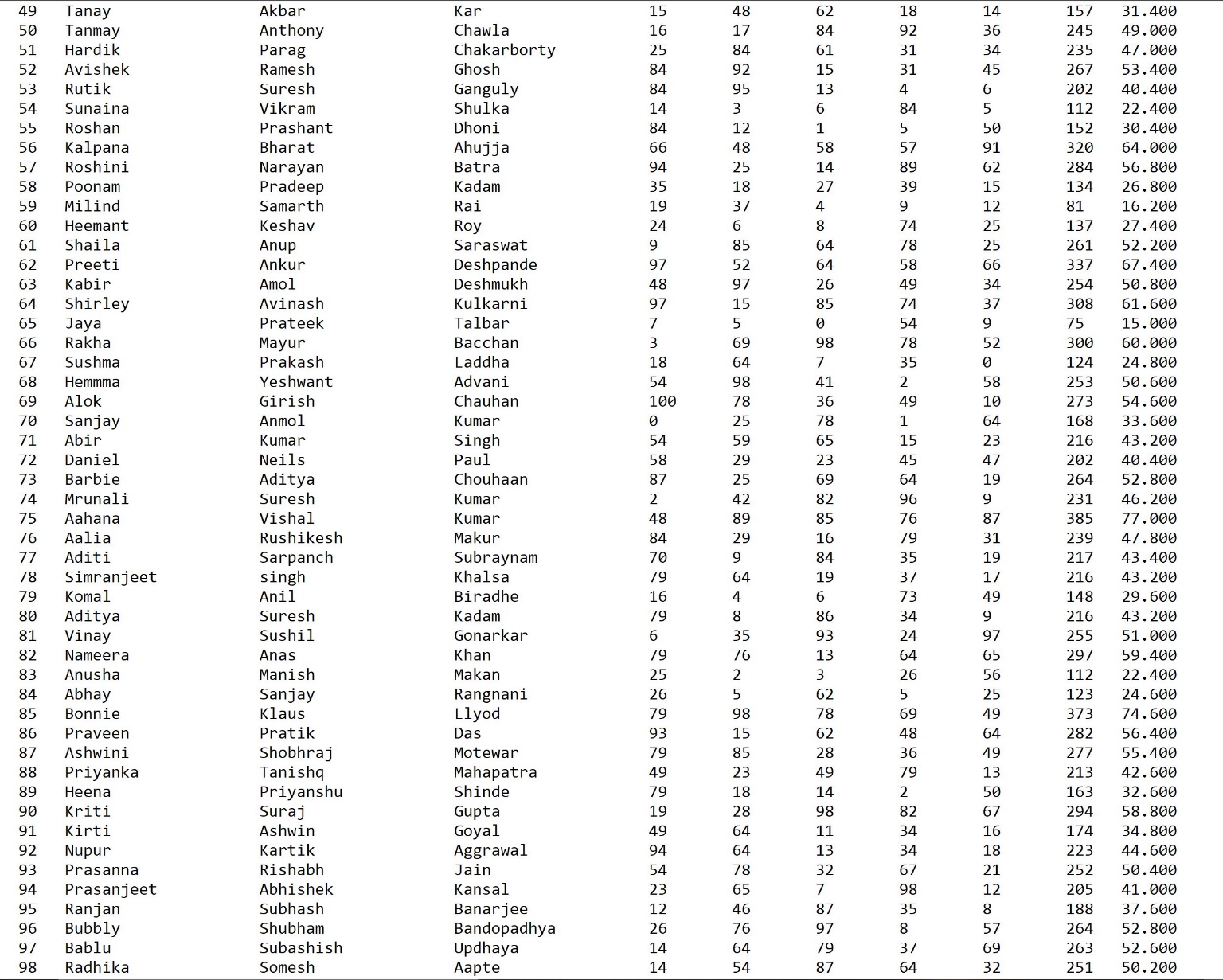


* OUTPUT (merit list prepared on marks 1):



For the sake of completeness the merit list on total marks of 100 students prepared by the developed software is given as below.





**END**