

Chapter 1

INTRODUCTION

In “STUDENT RECORD SYSTEM” we have done source coding, program testing and sorting the student record using quick sort technique. we are given the user the facility to enter the student record and sorting the record based on the key i.e. roll no.

Sorting is nothing but storage of data in sorted order, it can be in ascending or descending order. Sorting arranges data in a sequence which makes searching easier. Here Student roll no. can be taken as key for sorting the records in ascending or descending order. Now suppose we have to search a Student with roll no. 15, we don't need to search the complete record we will simply search between the Students with roll no 10 to 20. This application-project is an attempt to, making it easier for everyone to arrange data in an order, hence making it easier to search.

1.1 Motivation of project

"Student Records" are those records directly related to a student and maintained by entering the new student detail, sort the student's record, searching student record.

This project is to create a Student record system to store the roll no, sem, marks of different student using a linear data structure using c programming.

In this software one can easily add student's new record, sort student's record, search student's record, compute using algorithms and view all student records. This application stands out among all other software in a way that is user friendly and can be modified easily as per the requirements. It allows user to add/view/sort/search records.

SORTING THE STUDENTS RECORD

We can improve the efficiency of the system thus overcome the drawbacks of the existing system.

- Less human error
- Strength and strain of registers and papers can be reduced
- High security
- Data consistency
- Easy data updating
- Easy record keeping
- Backup data can be easily generated

Achieving this objective is difficult using manual system as the information is scattered, can be collecting relevant information may be very time consuming. All these problems are solved by using this project.

1.2 Problem statement

To design and implement of sorting the Student Record using quick sort technique that can overcome all the issues related.

Chapter 2

System Requirement Specification

Purpose: The purpose of the project is to maintain the student record by adding (the student details such as roll no, sem, marks), sorting the record in order to search the student detail easily.

The student Record may be chosen because it is thought to provide following advantages:

- 1.This software is space and time efficient.
- 2.There is a well-developed Document Management.
- 3.It is small and user friendly.
- 4.File Handling is effectively implemented.
- 5.Attractive design.

Scope: The application to finding a way to overcome the olden system of arranging the student record according to roll number or score wise to place them orderly. Data was kept unordered and unsorted, but fortunately the concept of sorting came into existence, making it easier for everyone to arrange data in an order, hence making it easier to search. Without a student Record managing and maintaining the details of student is unvarying.

2.1 Hardware System Configuration:

Processor	- Intel Core i5
Speed	- 1.8 GHz
RAM	- 256 MB (min)
Hard Disk	- 10 GB

2.2 Software System Configuration:

Operating System - windows 7/8.1/10

Programming Language - C

Compiler -Turbo C outcomes

Chapter 3

METHODOLOGY

3.1 Flowchart

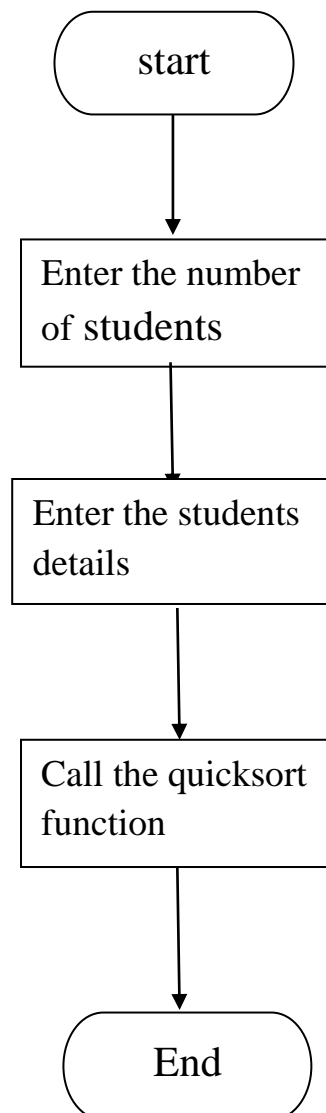


Fig no. 3.1. Flowchart for sorting the student record

It has the Student details included Roll.no, Sem, Marks for no. r of students. Then it calls the Quick sort function to sort the student record based on the Roll.no.

3.2 Algorithm

Step1: Start

struct student

{

int roll;

int sem;

int marks;

}

Step 2: function for quick sort

Declare variables i, j, l, h and declare temporary struct variable 'temp' for storing the temporary value.

- l=index of first element, h=index of last element
- assign the value of l to p and i, h to j.
- When $i < j$, if first element is \leq pivot then index of first element is incremented.
- And again, if last element is \geq pivot then the index value of last element is decremented.
- And then compare i and j, if $i < j$ swap s[i] and s[j].

Then according to the swapping, the student variables which are stored in s[i] goes to a temporary variable called temp[i], next the student variable stored in s[j] goes to a student variables of s[i] then student variables which are stored in a temporary variable called temp[i] goes to student variables of s[j].

- if $i > j$ then swap both s[j] and s[p].

Then according to the swapping, the student variables which are stored in s[j] goes to temporary variable called temp[i], next the student variables which are stored in s[p] goes to student variables of s[j] then student variables which are stored in temporary variable called temp[i] goes to student variables of s[p].

SORTING THE STUDENTS RECORD

- According to the above swapping the whole student record will arranged based on the sorting of roll.no.

Step 3: main function

- Enter the number of students.
- For $i=0$ to n , enter the details of student has roll number, sem, marks.
- For $i=0$ to n sort $s[i]$. roll.
- Then call the function quicksort.
- Print the sorted roll numbers.
- For $i=0$ to n , print the details of student according to the sorted roll numbers.

Step 4: End

3.3 Code and Implementation

```
#include<stdio.h>

struct student

{

int roll;

int sem;

int marks;

};

void quicksort (struct student s [], int l, int h)

{

int i, j, temp, p;

if(l<h)

{

p=l;

i=l;

j=h;

while(i<j)

{

while(s[i]. roll<=s[p]. roll&& i<=h)

{

i++;

}

}
```


SORTING THE STUDENTS RECORD

```
while(s[j]. roll>s[p]. roll)

j--;

if(i<j)

{

temp[i]. roll=s[i]. roll;

temp[i]. sem=s[i]. sem;

temp[i]. marks=s[i]. marks

s[i]. roll=s[j]. roll;

s[i]. sem=s[j]. sem;

s[i]. marks=s[j]. marks

s[j]. roll=temp[i]. roll;

s[j]. sem=temp[j]. sem;

s[j]. marks=temp[j]. marks;

}

}

temp[i]. roll=s[j]. roll;

temp[i]. sem=s[j]. sem;

temp[i]. marks=s[j]. marks;

s[j]. roll=s[p]. roll;

s[j]. sem=s[p]. sem;

s[j]. marks=s[p]. marks;

s[p]. roll=temp[i]. roll;
```

SORTING THE STUDENTS RECORD

```
s[p]. sem=temp[i]. sem;

s[p]. marks=temp[i]. marks;

quicksort (s, l, j-1);

quicksort (s, j+1, h);

}

}

void quicksort (struct student [], int l, int h);

int main ()

{

Struct student s [20], int i, j, n;

printf ("\nenter the number of students");

scanf ("%d", &n);

printf ("\nenter the roll, sem, marks");

for (i=0; i<n; i++)

{

scanf ("%d%d%d", &s[i]. roll, &s[i]. sem, &s[i]. marks);

}

for (i=0; i<n; i++)

{

quicksort (s, 0, n-1);

}

printf ("the sorted array is: ");
```

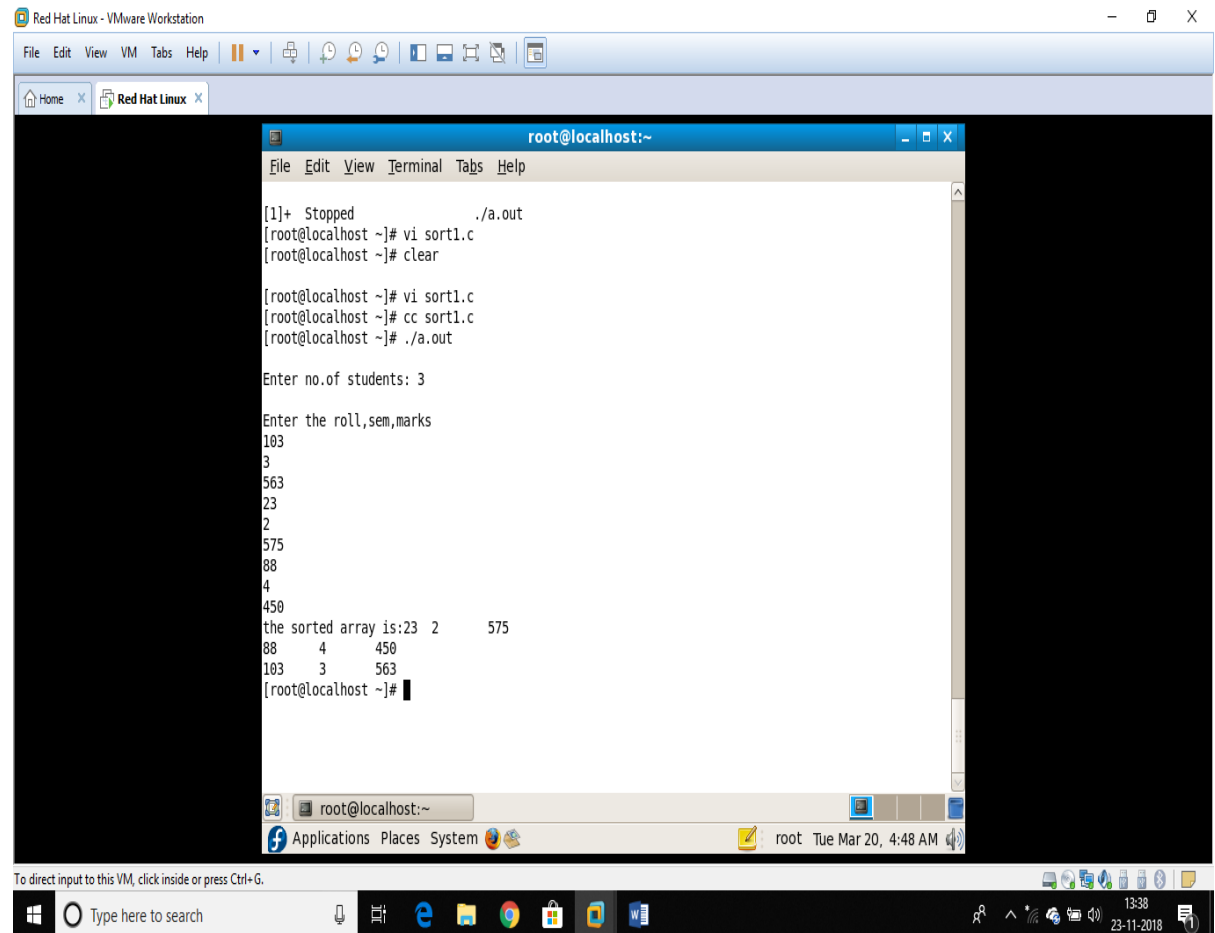
SORTING THE STUDENTS RECORD

```
for (i=0; i<n; i++)  
  
{  
  
printf ("%d\t%d\t%d\n", s[i]. roll, s[i]. sem, s[i]. marks);  
  
}  
  
return 0;  
  
}
```

Chapter 4

RESULTS AND DISCUSSION

4.1 Output (Snapshots)

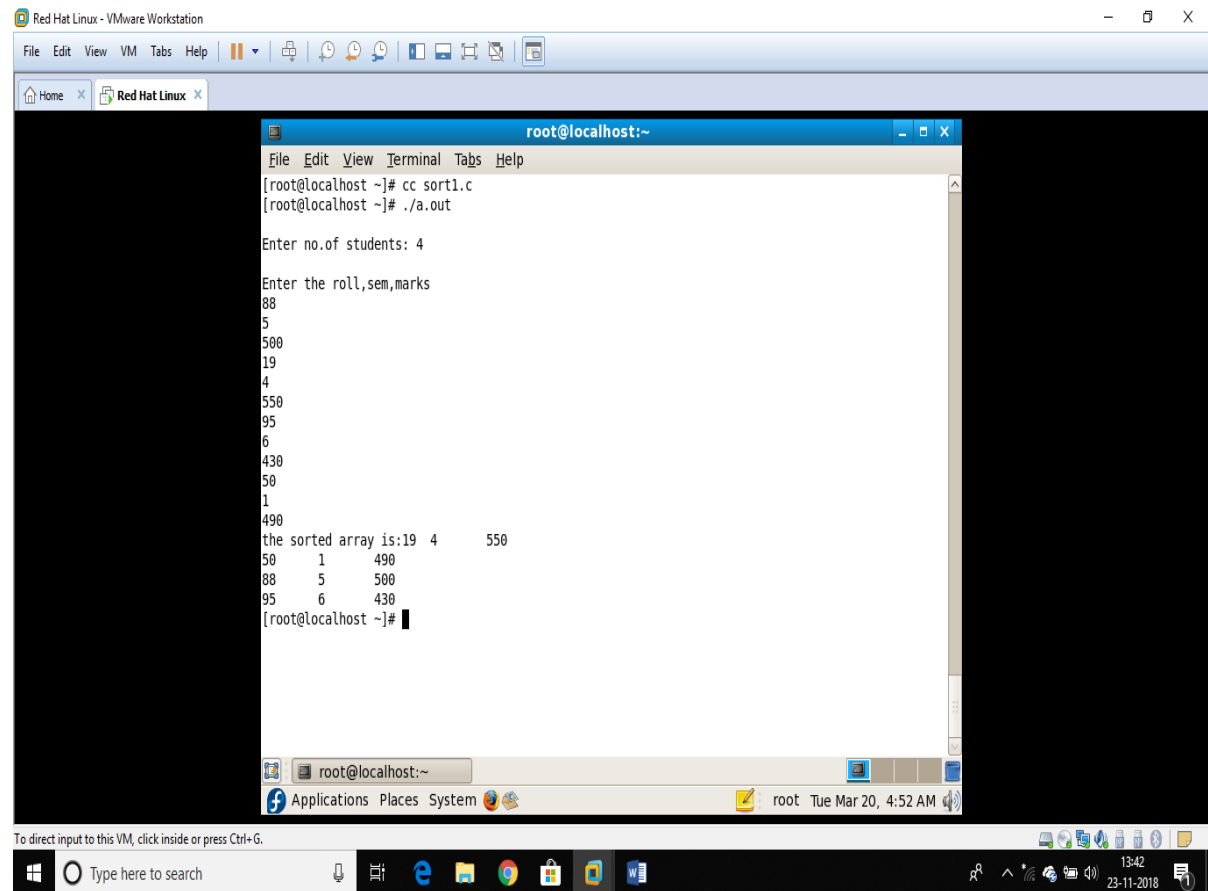


```
root@localhost:~  
File Edit View Terminal Tabs Help  
[1]+ Stopped ./a.out  
[root@localhost ~]# vi sort1.c  
[root@localhost ~]# clear  
  
[root@localhost ~]# vi sort1.c  
[root@localhost ~]# cc sort1.c  
[root@localhost ~]# ./a.out  
  
Enter no.of students: 3  
  
Enter the roll,sem,marks  
103  
3  
563  
23  
2  
575  
88  
4  
450  
the sorted array is:23 2 575  
88 4 450  
103 3 563  
[root@localhost ~]#
```

Fig no 4.1(Output snapshot 1: Ask to enter the no. of students.)

Then entering the student details are Roll.no, Sem, Marks. It displays the sorted student record based on the key Roll.no.

SORTING THE STUDENTS RECORD



```
root@localhost:~# cc sort1.c
root@localhost:~# ./a.out

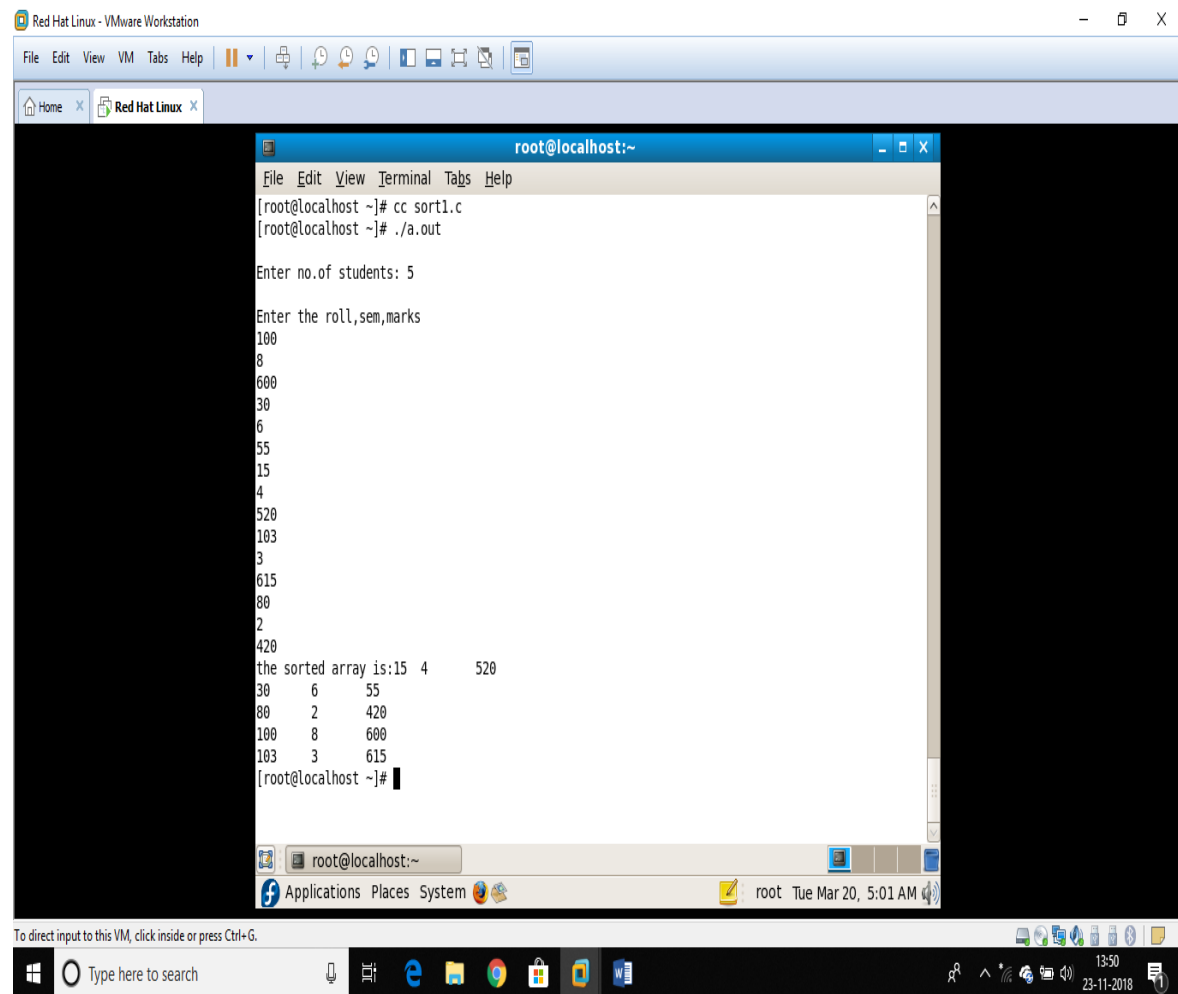
Enter no.of students: 4

Enter the roll,sem,marks
88
5
500
19
4
550
95
6
430
50
1
490
the sorted array is:19 4 550
50 1 490
88 5 500
95 6 430
root@localhost:~#
```

Fig no 4.2(Output snapshot 2: Ask to enter the no. of students.)

Then entering the student details are Roll.no, Sem, Marks. It displays the sorted student record based on the key Roll.no.

SORTING THE STUDENTS RECORD



The screenshot shows a Red Hat Linux terminal window titled 'root@localhost:~'. The user has compiled a C program named 'sort1.c' and executed it as './a.out'. The program prompts the user to 'Enter no. of students: 5' and then 'Enter the roll,sem,marks'. The user enters the following data:

Roll.no.	Sem.	Marks
100	8	600
30	6	55
15	4	520
103	3	615
80	2	420

The program then displays the sorted array based on roll number:

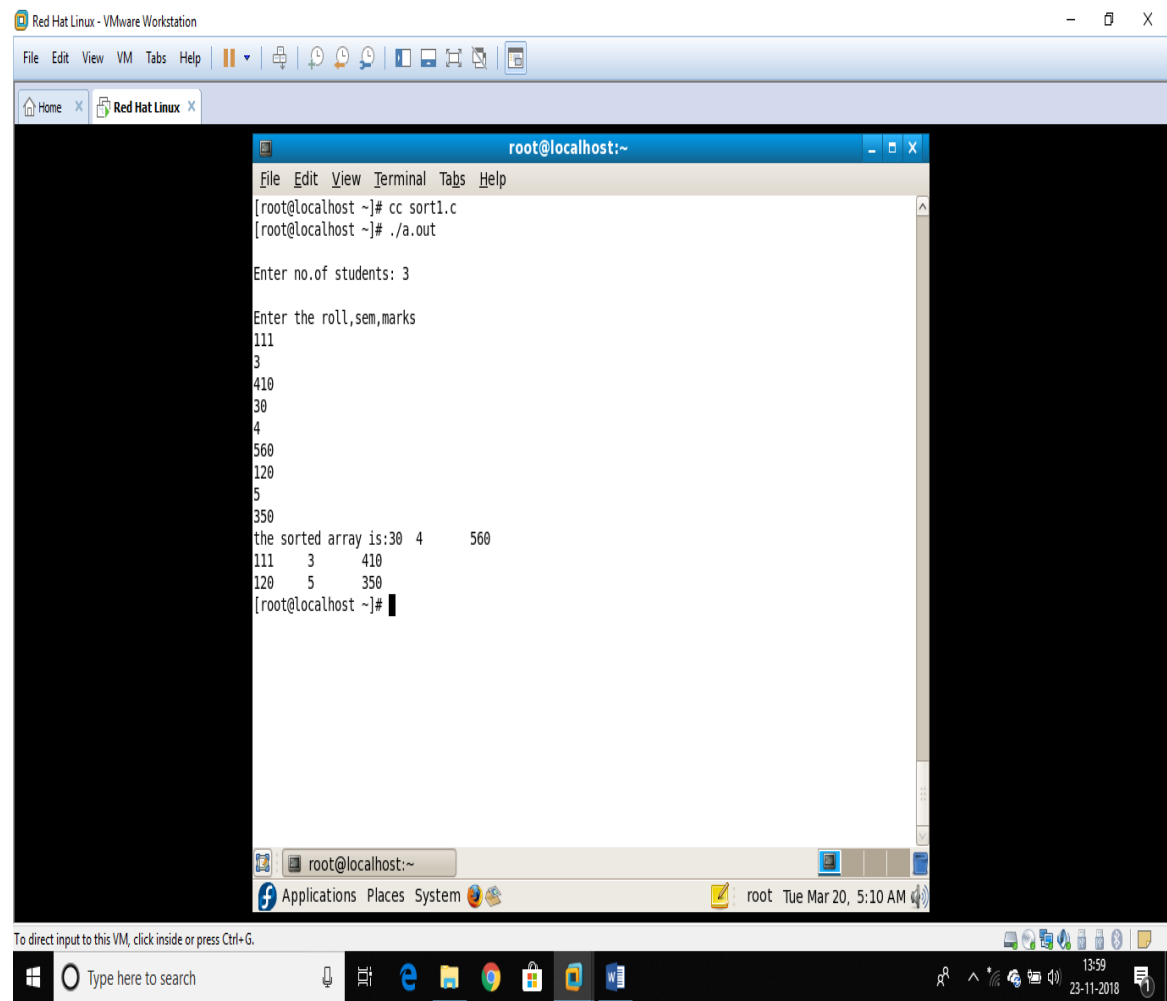
```
the sorted array is: 15 4 520
30 6 55
80 2 420
100 8 600
103 3 615
```

The terminal window is part of a VMware Workstation environment, as indicated by the title bar 'Red Hat Linux - VMware Workstation'. The bottom of the image shows the Windows taskbar with the search bar and system clock.

Fig no 4.3(Output snapshot 3: Ask to enter the no. of students.)

Then entering the student details are Roll.no, Sem, Marks. It displays the sorted student record based on the key Roll.no.

SORTING THE STUDENTS RECORD

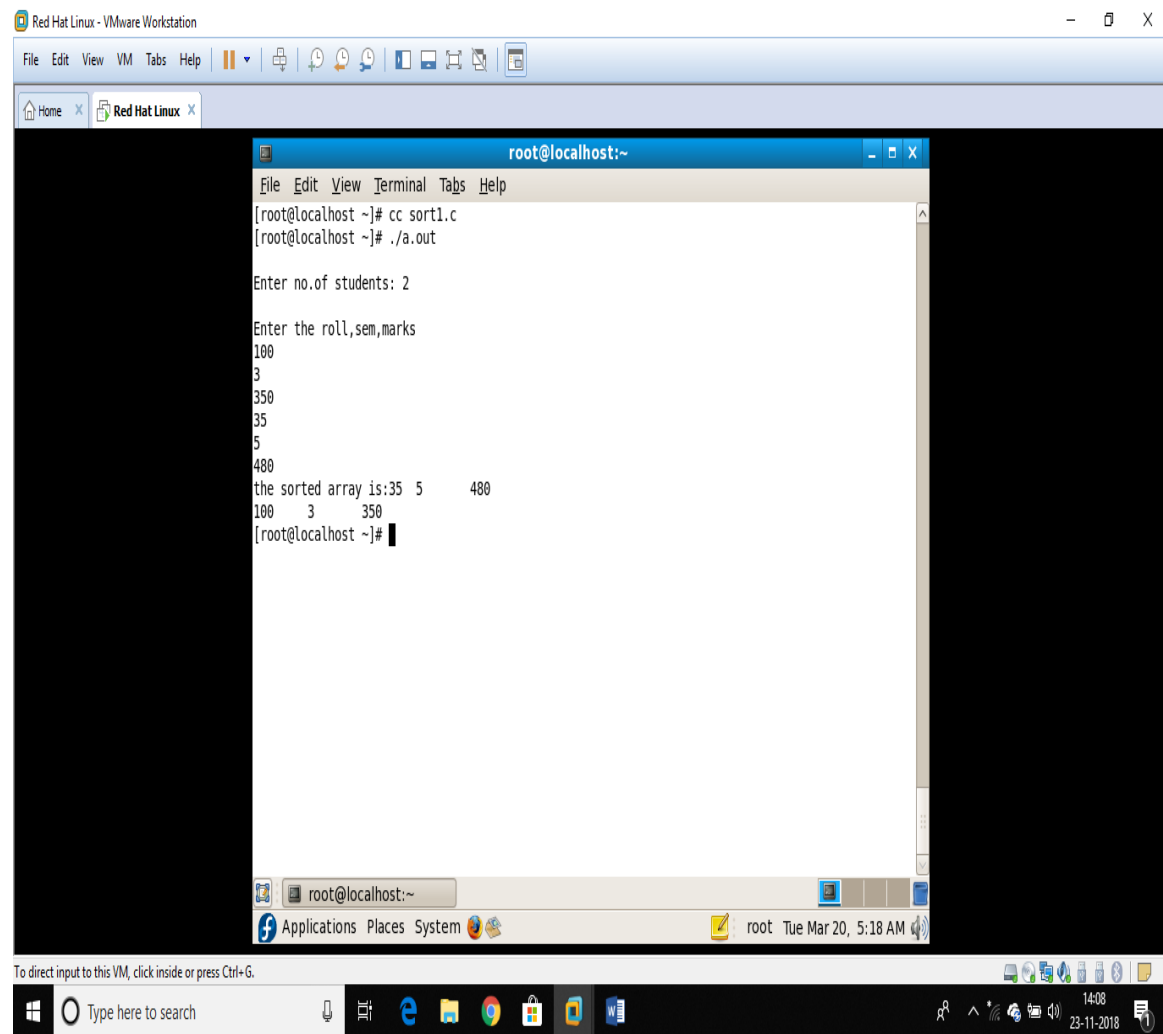


```
root@localhost:~  
File Edit View Terminal Tabs Help  
[root@localhost ~]# cc sort1.c  
[root@localhost ~]# ./a.out  
  
Enter no.of students: 3  
  
Enter the roll,sem,marks  
111  
3  
410  
30  
4  
560  
120  
5  
350  
the sorted array is:30 4 560  
111 3 410  
120 5 350  
[root@localhost ~]#
```

Fig no 4.4(Output snapshot 4: Ask to enter the no. of students.)

Then entering the student details are Roll.no, Sem, Marks. It displays the sorted student record based on the key Roll.no.

SORTING THE STUDENTS RECORD



The screenshot shows a Red Hat Linux - VMware Workstation window. Inside, a terminal window titled 'root@localhost:~' is open. The terminal displays the following text:

```
File Edit View Terminal Tabs Help
[root@localhost ~]# cc sort1.c
[root@localhost ~]# ./a.out

Enter no.of students: 2

Enter the roll,sem,marks
100
3
350
35
5
400
the sorted array is:35 5 400
100 3 350
[root@localhost ~]#
```

The terminal window is part of a desktop environment with a taskbar at the bottom showing various application icons and system status information (root Tue Mar 20, 5:18 AM).

Fig no 4.5(Output snapshot 5: Ask to enter the no. of students.)

Then entering the student details are Roll.no, Sem, Marks. It displays the sorted student record based on the key Roll.no.

CONCLUSION AND FUTURE ENHANCEMENT

This application-project reduces the manual work, maintaining accuracy, increasing efficiency and saving time.

It avoids the time taking works like comparing and arranging the things in desirable manner, and just by entering the random roll number, sem, names in the sorting programs, user can get desirable output anywhere and anytime.

References

www.itxperts.co.in

Ecomputersnotes.com

www.slideshare.net

www.Beginnersbook.com

www.geeksforgeeks.org

“Cracking the Coding Interview: 189 Programming Questions and Solutions” by Gayle Laakmann McDowell.

