G. H. Raisoni College Of Engineering And Management, Wagholi Pune

Assignment no :- 1 2021- 2022

Department	CE [SUMMER 2022 (Online)]		
Term / Section	III/B	Date Of **submission**	27-09-2021
Subject Name /Code	Data structure and algorithms / UCSL201 / UCSP201		
Roll No.	SCOB77	Name	Pratham Rajkumar pitty
Registration Number	2020AC0E1100107		

EXPERIMENT NO. 1

Aim - Consider a student database of SY comp class (at least 10 records). Database contains different fields of every student. Like Roll No. Name, and SGPA. (array of objects of class).

(a) Design a roll call list, arrange list of students according to roll numbers in asending order (Bubble Sort).

(b) Arrange List of Students alphabetically (use Incertion Sort).

(c) Arrange list of students to Find out first 10 topers from a class (use quick Sort)

(d) Searching Students according to SGPA. If more than ~~that~~ one student having same SGPA, then print list of all students having same SGPA.

(e) Search a particular student according to name using binary search without recursion

Objective ->

1) To Study The concepts of array structure.

2) Apply different sorting techiques on array of structure (Bubble, Insertion and quick sort)

3) display the output for every pass

4) Appy different sorting techniques on array of structure (Linear Search, Binary Search) and display output for every pass.

5) calculate time complexity.

Theory -

(1) Structure - Structure is collection of different data items in such a way that they can be reffered as single unit.

- Creating a Structure

```
struct student
{
int roll_no;
char name[15];
float sgpa;
};
```

Hear, Struct keyword used to delare structure Student is name of structure. In the example, threet types of variables int char float.

- Declaring Structure variable - After declaring structure, we can define structure variables. for instance, variables are defined with structure data type of automobile.

```
Struct student S1, S2, S3;
```

Hear,

variables defined by struct of student. suppose, to declare array of 20 elements we write

```
Struct students [20];
```

This creat 20 sets of variables that are organised.

Indeed, to access a specific structure.

index array name
If we want to print name of 3rd customr
we write
```
cout << S[2].name;
```

▶ Refering Structure members with Dot operation:

Given the struct student and S_1 is is varidble of type struct student we can access the fields in following ways:

```
S1 roll=no;
S1.name ;
S1 sgpa;
```

▶ pointer to structure:

We can declare a pointer variable as ~~Structo~~
struct student *pointer or
struct student *ptr;
And suppose if we have structure variable declared as;
struct ~~struc~~ student S_1;
Then ptr can store address by;
ptr = & S_1

▶ Refering a structure member with (->):

using ptr we can access fields of S_1 as
ptr -> name or (* ptr).name.

▶ Array of structure — To declare an array, we first define a structure and then declare array variables of that Type.

Sorting :→

(1) **Bubble Sort** :→ Bubble sort is simplest sorting algo method that works by repeteadly swapping adjacent elements if they are in wrong (improper) order.

It's avg and worst-case complexity of Bubble Sort is $O(n^2)$, where 'n' is ~~a natural~~ a no. of item

Working -

Step1- Starting with first element, compare current element with next element of array.

Step2. If current element is greater than next, swap them.

Step3 If current element is less than next, move to next element.

Step4 Repeat step 1, till list is sorted.

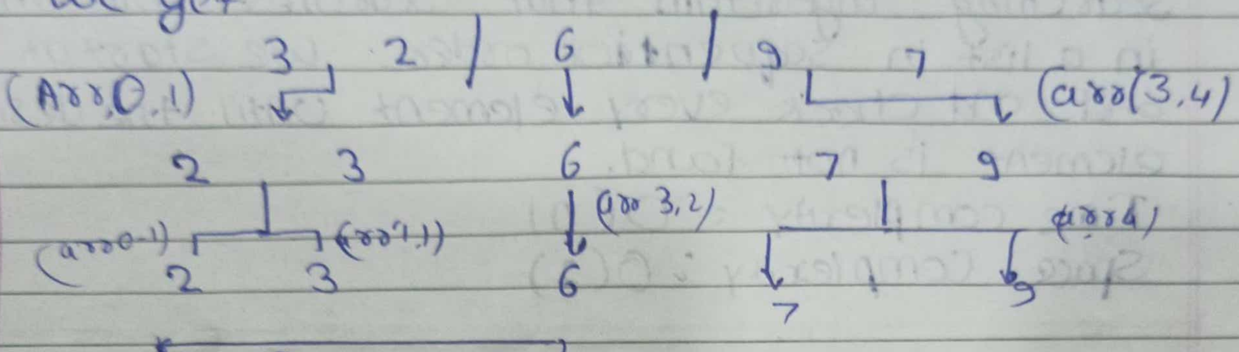
#2) **quick sort** - quick sort is algorithm based on divide and conquer approach in which array is split into sub-arrays and these sub arrays are recursively called sort element.

It's worst case complexity is $O(n^2)$ ~~Avg case~~ Avg. case is $O(n\log n)$. It occurs when pivot elements lies in extreme end of sorted array. one sub array is called also empty and another has $(n-1)$ elements

example - consider an array as
arr[5] = {7, 9, 3, 2, 6}

0	1	2	3	4
7	9	3	2	6

we consider pivot elements as 6 (last element)
so and after we write Q S(arr, 0, 4)
we get



{2, 3, 6, 7, 9}

3) Insertion Sort

Insertion sort is simple sorting algorithm that working the way we sort playing cards in our hands. If we assume that the ~~Insertion Sort~~ first card is already sorted then, we select an unsorted card. If the unsorted card is greater than the card in hand. it is placed on the right otherwise, to the left

Working :->

Step 1 -> pick next element
Step 2 -> compare with all elements in sorted sublist. (on left)
Step 3 -> Shift all elements in sorted sublist that is ~~ops~~ greater than value to be sort.
Step 4 -> Insert value
Step 5 -> Repeat until list is sorted.

Searching ↳ It is a process of retrieving or Locating a record with a particular key value.

- Linear Search →
Linear Search is the simplest searching algorithm that searches for an element in a list in Sequential order. We start at one end and Check every element until the desired element is not found.
Time complexity : O(n)
Space complexity : O(1)

- Binary search - Binary search is the Search techque which works effiently on the sorted lists Hence, in order to search an element into some list by using binary Search Technique, we must ensure that the list is sorted.

Binary Search follows divide and conquer approach in which, the list is divided into two halves and the item is compared with the middle element of the list if the match is found then, the location of middle element is returned otherwise, we search into either of the halves depending upon the result produced through the match.

ALGORITHM / PSEUDOCODE :

➢ **Create a structure**

Create_database(struct student s[])

```
Step 1: Accept how many records user need to add, say, no of records as n
Step 2: For i = 0 to n – 1
    i. Accept the record and store it in s[i]
Step 3: End For
Step 4: stop
```

Display_database(struct student s[] ,int n)

```
Step 1: For i = 0 to n –1
    i. Display the fields  s.roll_no, s.name, s.sgpa
Step 2: End For
Step 3: Stop
```

Bubble Sort student according to sort to roll numbers

```
BubbleSort(Student s[], n)
Step 1: For Pass = 1 to n-1
Step 2: For i = 0 to (n – pass – 1)
    i. If s[i].roll_no < s[i+1].roll_no
20
        a. Swap (s[i]. s[i+1])
    iii. End if
Step 3: End for
Step 4: End For
Step 5 :Stop
```

➢ **Insertion Sort to sort student on the basis of names**

insertion_Sort (Struct student S[], int n)

```
Step 1: For i = 1 to n-1
    i. Set key to s[i]
    ii. Set j to i-1
    iii. While j>=0 AND strcmp(s[i].name,key.name)>0
        a. Assign s[j] to s[j+1]
        b. Decrement j
    iv. End While
Step 2: Assign key to s[j+1]
Step 3: End for
Step 4: end of insertion sort
```

➢ **Quick Sort to sort students on the basis of their sgpa.**

partition (struct student s[], int l, int h)

```
// where s is the array of structure , l is the index of starting element
// and h is the index of last element.
Step 1: Select s[l].sgpa as the pivot element
Step 2: Set i = l
Step 3: Set j = h-1
Step 4: While i ≤ j
    i. Increment i till s[i].sgpa ≤ pivot element
    ii. Decrement j till s[j].sgpa > pivot element
    iii. If i < j
    iv. Swap(s[i], s[j])
    v. End if
Step 5:End while
Step 6: Swap(s[j],s[l])
Step 7: return j
Step 8 :end of Partition
```

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

```
//where s is the array of structure , l is the index of starting element
//and h is the index of last element.
Step 1: If l<h
    i. P=partition(s,l,h)
    ii. quicksort (s,l,p-1)
    iii. quicksort (s,p+1,h)
Step 2: End if
Step 3: end of quicksort
```

21

- **Algorithm for Line ar Search to search students with sgpa given and display all of them**

Linear_search (struct student s[], float key, int n)

```
//Here s is array of structure student, key is sgpa of student to be searched and
// displayed, n is total number of students in record
Step 1: Set i to 0 and flag to 0
Step 2: While i<n
    i. If s[i].sgpa==key
        a. Print s[i].roll_no, s[i].name
        b. Set flag to 1
        c. i++
Step 3: End while
Step 4: If flag==0
    i. Print No student found with sgpa=value of key
Step 5: End if
Step 6: End of linear_ sear
```

- **Algorithm for Binary Search to search students having given string in their names**

Binary_Search (s, n , Key)

```
// Where s is an array of structure , n is the no of records, and key is element to be searched
Step 1: Set l = 0 & h = n-1
Step 2:While l ≤ h
    i. mid = (l + h ) / 2
    ii. If strcmp (s[mid].name, key)==0)
        a. foun
        b. stop
    iii. Else
        a. if (strcmp (key, s[mid].name)<0
            i. h = mid – 1
        b. Else
            ii. l = mid + 1
```

c. End if
iv. End if
Step 4: End while
Step 5: not found // search is unsuccessful

Validation:

- Limit of the array should not be –ve, and should not cross the lower and upper bound.
- Roll numbers should not repeat, should not -ve
- Name should only contain alphabets, space and .
- Should not allowed any other operations before the input list is entered by the user .
- Before going to (binary search) records should be sorted according a names.

Test Cases :

- **Sorting :**

Four test cases :
22
i. Already Sorted according to the requirement
ii. Sorted in reverse order
iii. Partially sorted
iv. Completely Random List

Expected output /analysis is :
i. Test algorithm for above four test cases
ii. Analyze the algorithms based on no of comparisons and swapping/shifting required
iii. Check for Sort Stability factor
iv. No of passes needed
v. Best /average/ worst case of the each algorithm based on above test case
vi. Memory space required to sort

Searching :

- Find the max and minimum comparison required to search element
- Calculate how many comparisons are required for unsuccessful search

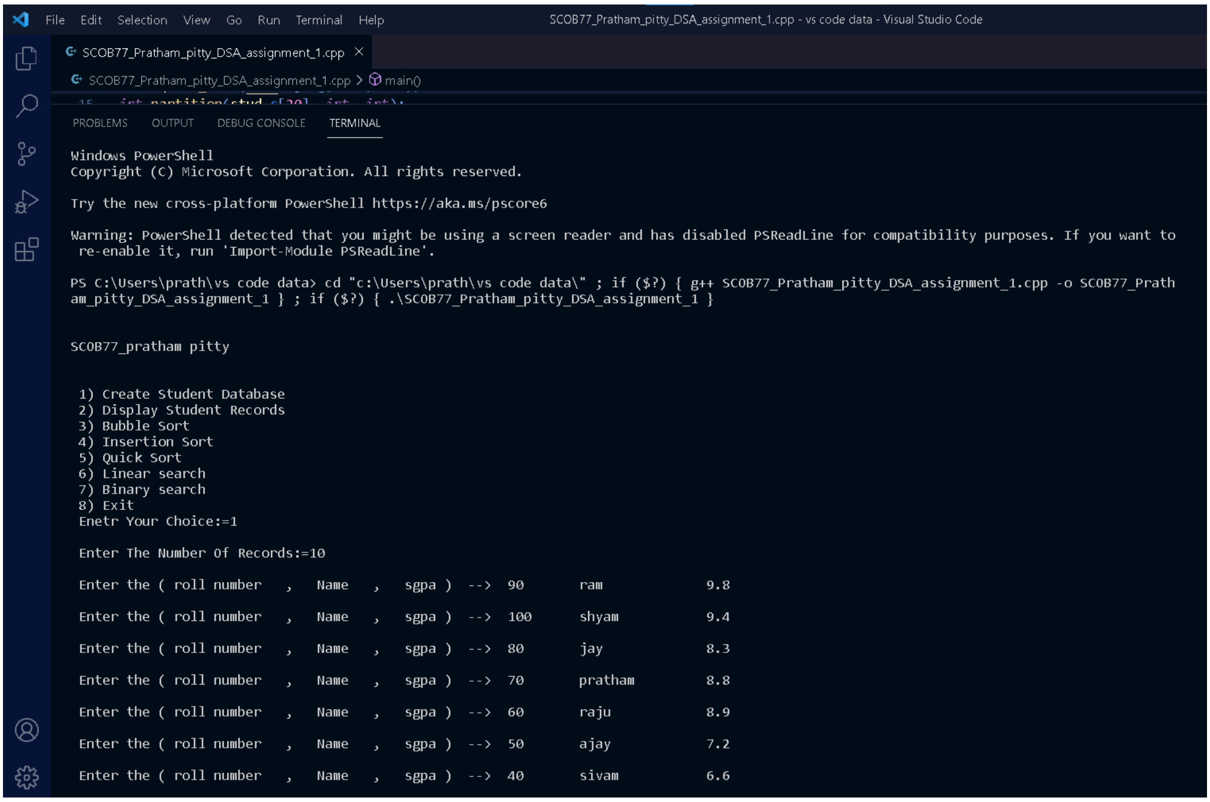
Application:
Useful in managing large amount of data.

Program code

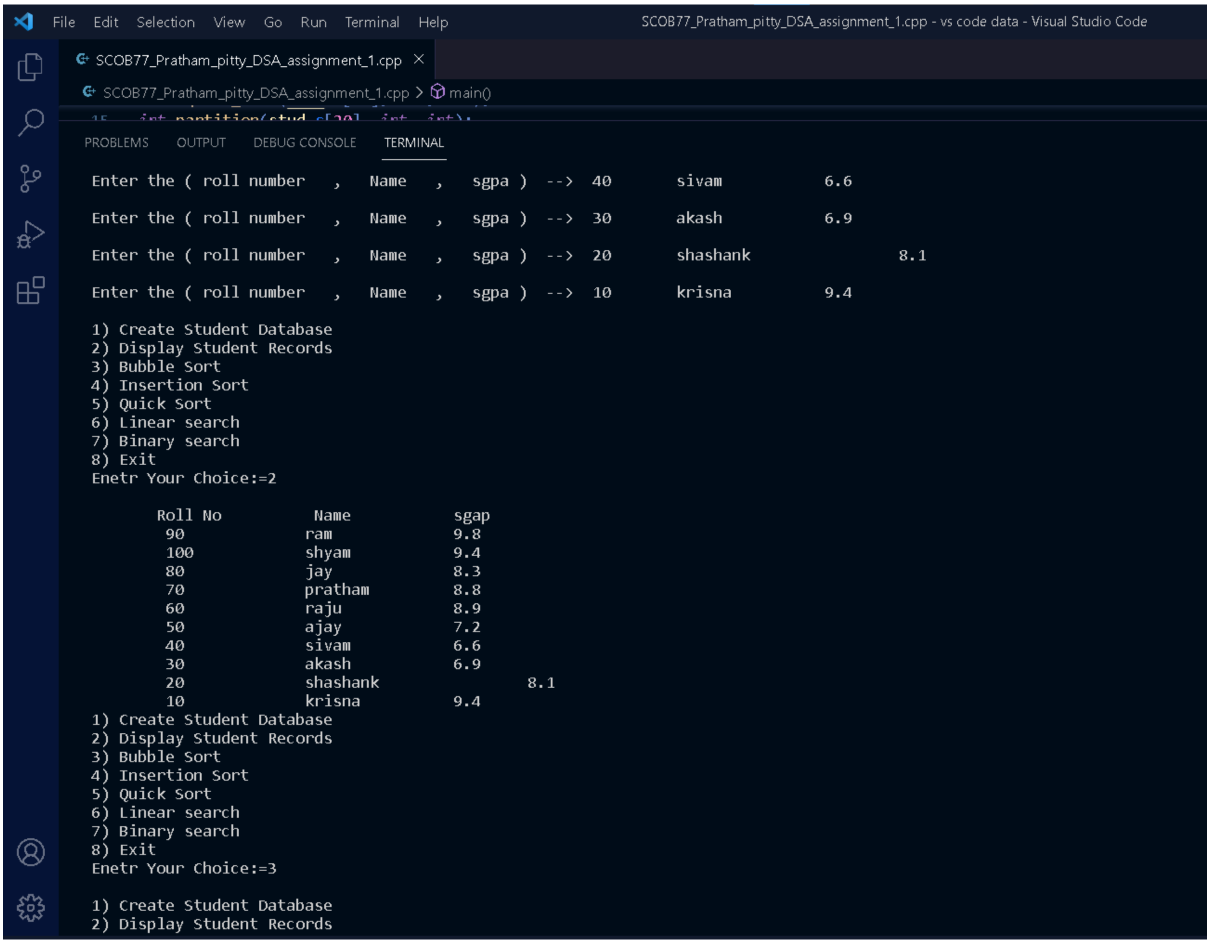
```
#include <iostream>

#include <string.h>

using namespace std;

typedef struct student

{

   int roll_num;

   char name[20];

   float sgpa;

} stud;

void create(stud s[20], int n);

void display(stud s[20], int n);

void bubble_sort(stud s[20], int n);

void insertionSort(stud s[20], int n);

void quick_sort(stud s[20], int, int);

int partition(stud s[20], int, int);

void search(stud s[20], int n, float key);

int bsearch(stud s[20], char x[20], int low, int high);


int main()

{

   stud s[20];

   int ch, n, result;

   float key;

   char x[20];

   cout<<"\n\nSCOB77_pratham pitty\n\n";

   do
{
   cout << "\n 1) Create Student Database ";
   cout << "\n 2) Display Student Records ";
   cout << "\n 3) Bubble Sort ";
   cout << "\n 4) Insertion Sort ";
   cout << "\n 5) Quick Sort ";
   cout << "\n 6) Linear search ";
   cout << "\n 7) Binary search  ";
   cout << "\n 8) Exit ";
   cout << "\n Enetr Your Choice:=";
   cin >> ch;
   switch (ch)
   {
   case 1:
      cout << "\n Enter The Number Of Records:=";
      cin >> n;
      create(s, n);
      break;
   case 2:
      display(s, n);
      break;
   case 3:
      bubble_sort(s, n);
      break;
   case 4:
      insertionSort(s, n);
```

```
    break;
  case 5:
    quick_sort(s, 0, n - 1);
     cout<<"\n"<< "\t"<< "Roll No"<< "\t\t"<<" Name" <<"\t\t"<< "sgap";
    for (int i = n - 1; i >= n - 10; i--)
    {
      cout << "\n";
      cout << "\t " << s[i].roll_num << "\t\t " << s[i].name << "\t\t " << s[i].sgpa;
    }
    break;
  case 6:
    cout << "\n Enter the sgpa which u want to search:=";
    cin >> key;
    search(s, n, key);
    break;
  case 7:
    cout << "\n Enter the name of student  which u want to search:=";
    cin >> x;
    insertionSort(s, n);
    result = bsearch(s, x, 0, (n - 1));
    if (result == -1)
    {
      cout << " \n Student name you want to search for is not present ! \n";
    }
    else
    {
            cout << " \n The student  is present :\t" << s[result].name;

          }

          break;

        case 8:

          return 0;

        default:

          cout << "\n Invalid choice !! Please enter your choice again." << endl;

        }

    } while (ch != 8);

}


void create(stud s[20], int n)

{

    int i;

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

    {

        cout<<"\n Enter the ( roll number   ,   Name   ,   sgpa )  -->  ";

        cin>>s[i].roll_num>>s[i].name>>s[i].sgpa;

    }

}

void display(stud s[20], int n)

{

    int i;

    cout<<"\n"<< "\t"<< "Roll No"<< "\t\t"<<" Name" <<"\t\t"<< "sgap";

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

    {
        cout << "\n";
        cout << "\t " << s[i].roll_num << "\t \t" << s[i].name << "\t \t" << s[i].sgpa;
    }
}
//bubble sort to sort in ascending order on roll number


void bubble_sort(stud s[20], int n)
{
    int i, j;
    stud temp;
    for (i = 1; i < n; i++)
    {
        for (j = 0; j < n - i; j++)
        {
            if (s[j].roll_num > s[j + 1].roll_num){
                temp = s[j];
                s[j] = s[j + 1];
                s[j + 1] = temp;
            }
        }
    }
}
// insertion sort to sort on names in ascending order
void insertionSort(stud s[20], int n)
{
    int i, j;
    stud key;
    for (i = 1; i < n; i++)
    {
        key = s[i];
        j = i - 1;
        /* Move elements of arr[0..i-1], that are
        greater than key, to one position ahead
        of their current position */
        while (j >= 0 && strcmp(s[j].name, key.name) > 0)
        {
            s[j + 1] = s[j];
            j = j - 1;
        }
        s[j + 1] = key;
    }
}

//Quick sort to sort on sgpa
void quick_sort(stud s[20], int l, int u)
{
    int j;
    if (l < u)
    {
        j = partition(s, l, u);
        quick_sort(s, l, j - 1);
        quick_sort(s, j + 1, u);
    }
}
int partition(stud s[20], int l, int u)
{
    int i, j;
    stud temp, v;
    v = s[l];
    i = l;
    j = u + 1;
    do
    {
        do
            i++;

        while (s[i].sgpa < v.sgpa && i <= u);

        do
            j--;
        while (v.sgpa < s[j].sgpa);

        if (i < j)
        {
            temp = s[i];
            s[i] = s[j];
            s[j] = temp;
        }
    } while (i < j);


    s[l] = s[j];
    s[j] = v;


    return (j);
}
// linear search for sgpa if more than one student having same sgpa print all of them
void search(stud s[20], int n, float key)
{
    int i; int c=0;
    cout<<"\n"<< "\t"<< "roll no"<< "\t\t"<<" name" <<"\t\t"<< "sgpa";
    for (i = 0; i < n; i++)
    {
        if (key == s[i].sgpa)
        {
            cout << "\n\t " << s[i].roll_num << "\t\t " << s[i].name << "\t\t " << s[i].sgpa;
            c++;
        }
    }
    if (c==0)
    {
        cout<<"No record found for this sgpa value";
    }


}
```

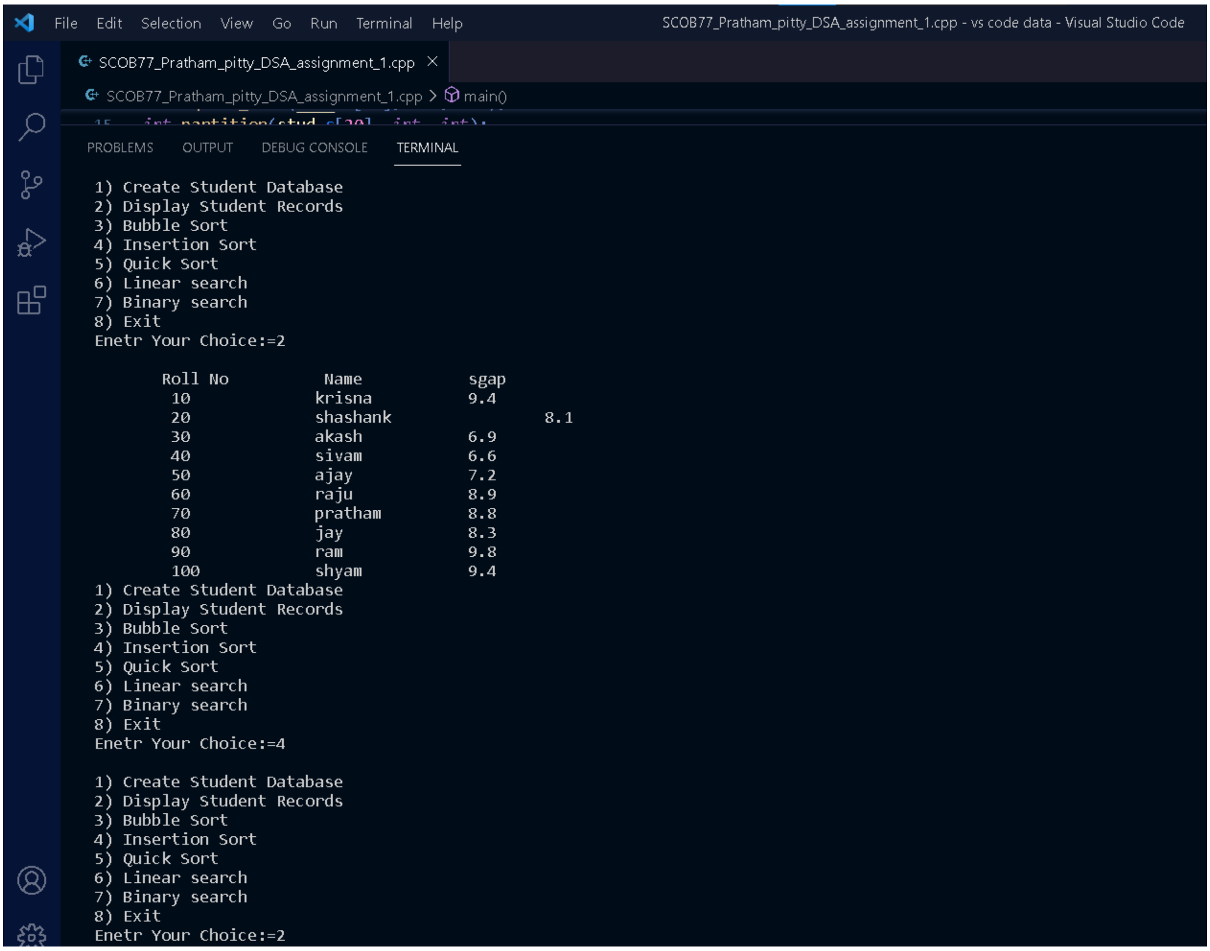
```
int bsearch(stud s[20], char x[20], int low, int high)
{
   int mid;
   while (low <= high)
   {
      mid = (low + high) / 2;
      if (strcmp(x, s[mid].name) == 0)
      {
         return mid;
      }
      else if (strcmp(x, s[mid].name) < 0)
      {
         high = mid - 1;
      }
      else
      {
         low = mid + 1;
      }
   }
   return -1;
}
```


File
Edit
Selection
View
Go
Run
Terminal
Help
SCOB77_Pratham_pitty_DSA_assignment_1.cpp - vs code data - Visual Studio Code
SCOB77_Pratham_pitty_DSA_assignment_1.cpp
SCOB77_Pratham_pitty_DSA_assignment_1.cpp > main()
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL

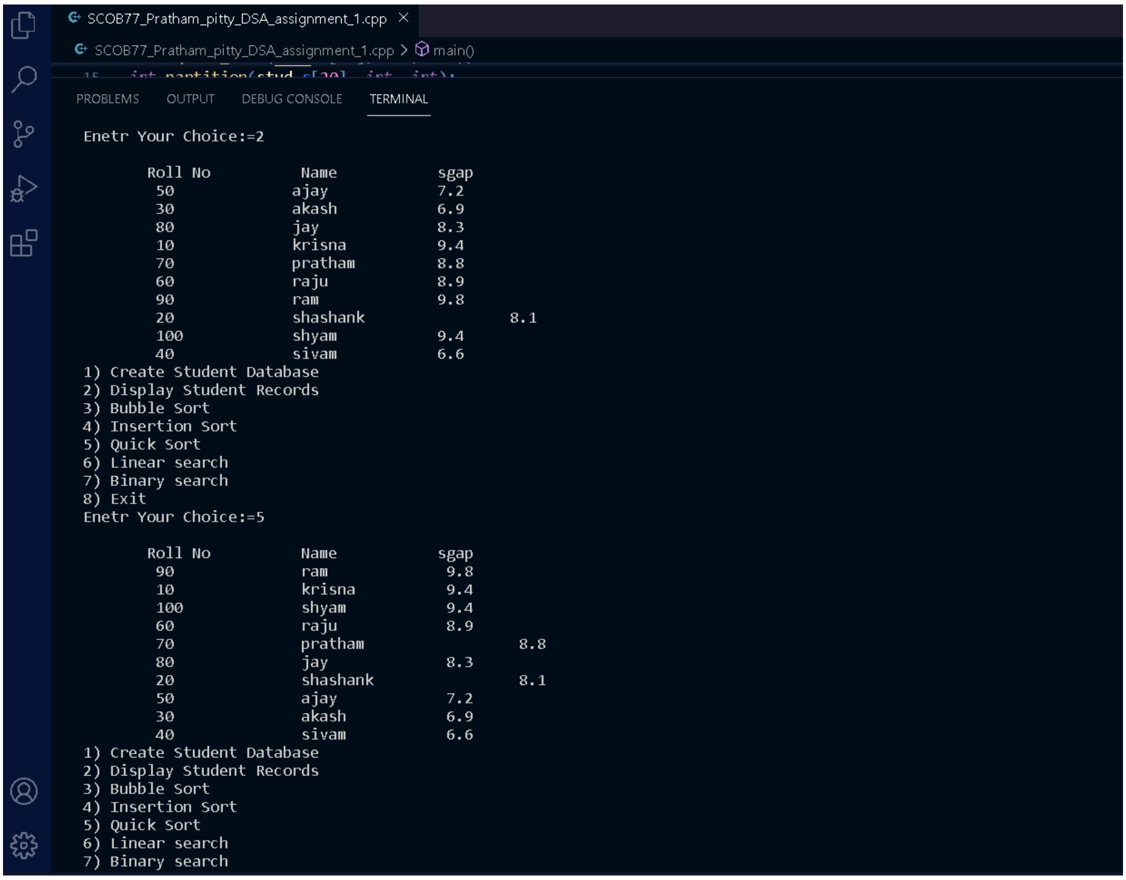
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to
 re-enable it, run 'Import-Module PSReadLine'.

PS C:\Users\prath\vs code data> cd "c:\Users\prath\vs code data\" ; if ($?) { g++ SCOB77_Pratham_pitty_DSA_assignment_1.cpp -o SCOB77_Prath
am_pitty_DSA_assignment_1 } ; if ($?) { .\SCOB77_Pratham_pitty_DSA_assignment_1 }


SCOB77_pratham pitty


 1) Create Student Database
 2) Display Student Records
 3) Bubble Sort
 4) Insertion Sort
 5) Quick Sort
 6) Linear search
 7) Binary search
 8) Exit
 Enetr Your Choice:=1

 Enter The Number Of Records:=10

 Enter the ( roll number    ,  Name    ,  sgpa )  -->  90       ram             9.8

 Enter the ( roll number    ,  Name    ,  sgpa )  -->  100      shyam           9.4

 Enter the ( roll number    ,  Name    ,  sgpa )  -->  80       jay             8.3

 Enter the ( roll number    ,  Name    ,  sgpa )  -->  70       pratham         8.8

 Enter the ( roll number    ,  Name    ,  sgpa )  -->  60       raju            8.9

 Enter the ( roll number    ,  Name    ,  sgpa )  -->  50       ajay            7.2

 Enter the ( roll number    ,  Name    ,  sgpa )  -->  40       sivam           6.6
```


File
Edit
Selection
View
Go
Run
Terminal
Help
SCOB77_Pratham_pitty_DSA_assignment_1.cpp - vs code data - Visual Studio Code
SCOB77_Pratham_pitty_DSA_assignment_1.cpp
SCOB77_Pratham_pitty_DSA_assignment_1.cpp > main()
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL

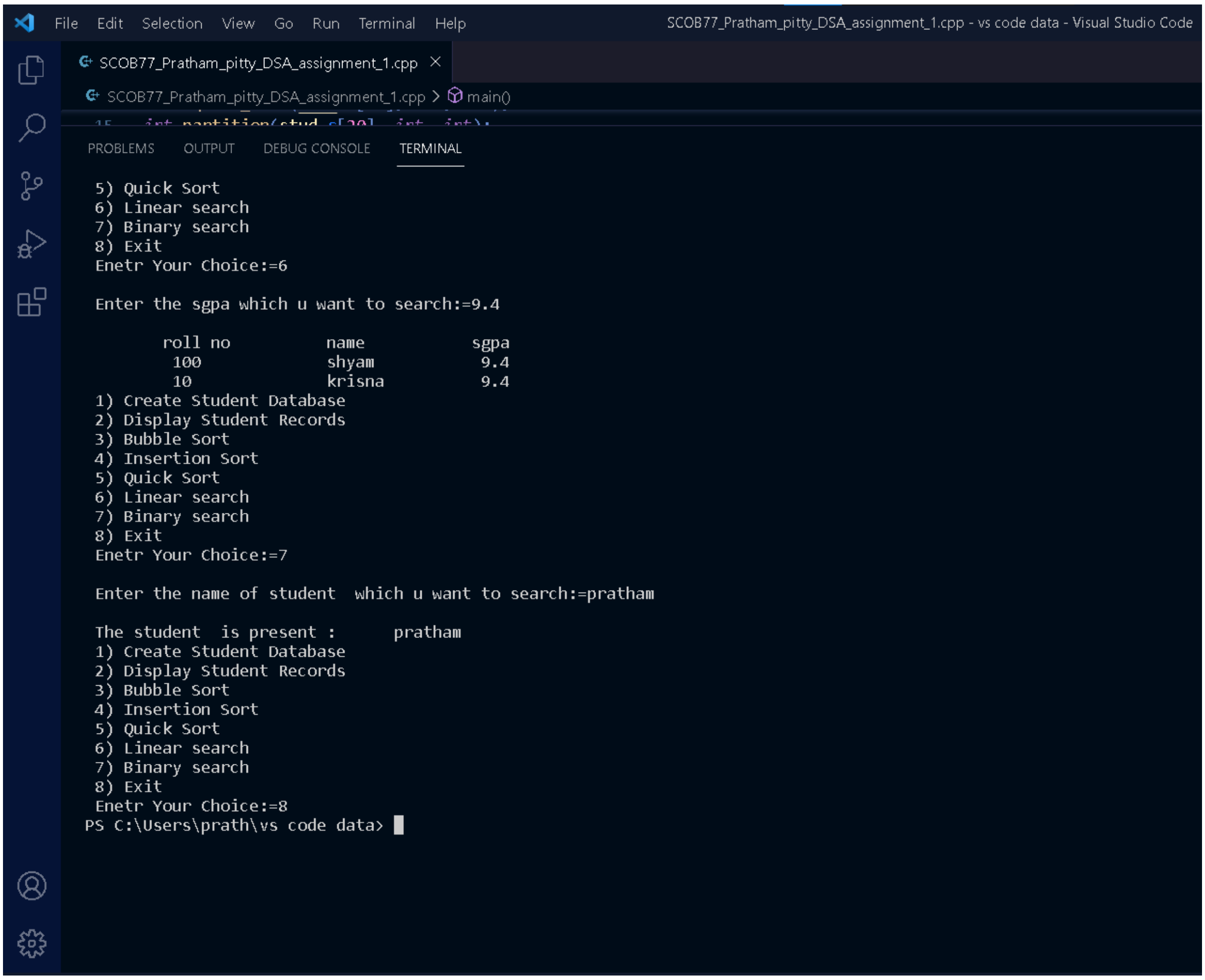
```
 Enter the ( roll number    ,  Name    ,  sgpa )  -->  40       sivam           6.6

 Enter the ( roll number    ,  Name    ,  sgpa )  -->  30       akash           6.9

 Enter the ( roll number    ,  Name    ,  sgpa )  -->  20       shashank                8.1

 Enter the ( roll number    ,  Name    ,  sgpa )  -->  10       krisna          9.4

 1) Create Student Database
 2) Display Student Records
 3) Bubble Sort
 4) Insertion Sort
 5) Quick Sort
 6) Linear search
 7) Binary search
 8) Exit
 Enetr Your Choice:=2

        Roll No          Name           sgap
         90             ram             9.8
         100            shyam           9.4
         80             jay             8.3
         70             pratham         8.8
         60             raju            8.9
         50             ajay            7.2
         40             sivam           6.6
         30             akash           6.9
         20             shashank                8.1
         10             krisna          9.4
 1) Create Student Database
 2) Display Student Records
 3) Bubble Sort
 4) Insertion Sort
 5) Quick Sort
 6) Linear search
 7) Binary search
 8) Exit
 Enetr Your Choice:=3

 1) Create Student Database
 2) Display Student Records
```


File
Edit
Selection
View
Go
Run
Terminal
Help
SCOB77_Pratham_pitty_DSA_assignment_1.cpp - vs code data - Visual Studio Code
SCOB77_Pratham_pitty_DSA_assignment_1.cpp
SCOB77_Pratham_pitty_DSA_assignment_1.cpp > main()
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL

```
1) Create Student Database
2) Display Student Records
3) Bubble Sort
4) Insertion Sort
5) Quick Sort
6) Linear search
7) Binary search
8) Exit
Enetr Your Choice:=2

       Roll No          Name           sgap
        10             krisna          9.4
        20             shashank                8.1
        30             akash           6.9
        40             sivam           6.6
        50             ajay            7.2
        60             raju            8.9
        70             pratham         8.8
        80             jay             8.3
        90             ram             9.8
        100            shyam           9.4
1) Create Student Database
2) Display Student Records
3) Bubble Sort
4) Insertion Sort
5) Quick Sort
6) Linear search
7) Binary search
8) Exit
Enetr Your Choice:=4

1) Create Student Database
2) Display Student Records
3) Bubble Sort
4) Insertion Sort
5) Quick Sort
6) Linear search
7) Binary search
8) Exit
Enetr Your Choice:=2
```


SCOB77_Pratham_pitty_DSA_assignment_1.cpp
SCOB77_Pratham_pitty_DSA_assignment_1.cpp > main()
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL

```
Enetr Your Choice:=2

       Roll No          Name           sgap
        50             ajay            7.2
        30             akash           6.9
        80             jay             8.3
        10             krisna          9.4
        70             pratham         8.8
        60             raju            8.9
        90             ram             9.8
        20             shashank                8.1
        100            shyam           9.4
        40             sivam           6.6
1) Create Student Database
2) Display Student Records
3) Bubble Sort
4) Insertion Sort
5) Quick Sort
6) Linear search
7) Binary search
8) Exit
Enetr Your Choice:=5

       Roll No          Name           sgap
        90              ram             9.8
        10              krisna          9.4
        100             shyam           9.4
        60              raju            8.9
        70              pratham                 8.8
        80              jay             8.3
        20              shashank                8.1
        50              ajay            7.2
        30              akash           6.9
        40              sivam           6.6
1) Create Student Database
2) Display Student Records
3) Bubble Sort
4) Insertion Sort
5) Quick Sort
6) Linear search
7) Binary search
```


File
Edit
Selection
View
Go
Run
Terminal
Help
SCOB77_Pratham_pitty_DSA_assignment_1.cpp - vs code data - Visual Studio Code
SCOB77_Pratham_pitty_DSA_assignment_1.cpp
SCOB77_Pratham_pitty_DSA_assignment_1.cpp > main()
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL

```
int partition(stud s[20], int, int);
```

```
 5) Quick Sort
 6) Linear search
 7) Binary search
 8) Exit
 Enetr Your Choice:=6

 Enter the sgpa which u want to search:=9.4

        roll no         name           sgpa
         100            shyam           9.4
         10             krisna          9.4
 1) Create Student Database
 2) Display Student Records
 3) Bubble Sort
 4) Insertion Sort
 5) Quick Sort
 6) Linear search
 7) Binary search
 8) Exit
 Enetr Your Choice:=7

 Enter the name of student  which u want to search:=pratham

 The student  is present :      pratham
 1) Create Student Database
 2) Display Student Records
 3) Bubble Sort
 4) Insertion Sort
 5) Quick Sort
 6) Linear search
 7) Binary search
 8) Exit
 Enetr Your Choice:=8
PS C:\Users\prath\vs code data> 
```