Name : Aritra Das

Roll No: 75

Assignment 7

1.

//Create an array of student objects (containing roll, name, name and score)

//whose size may vary dynamically once objects are added or removed, randomly

//elements may be accessed, one can find number of objects in the list, one can

//find the student with highest score, find the students with a substring in their

//name and also without a substring in the name. Take the help of suitable STL

//classes.

#include<iostream>

#include<string>

#include<vector>

using namespace std;

class Student{

    public:

    int roll,score;

    string name;

    Student(){

        cout<<"Enter roll, name, score:\n";

        cin>>roll>>name>>score;

    }

    void show(){

        cout<<"Roll : "<<roll<<" Name : "<<name<<" Score : "<<score<<endl;

    }

};

void highScore(vector<Student> arr){

    int max=-10000;

    for(auto it = arr.begin();it!=arr.end();++it)

        if(it->score>max)

            max = it->score;

    for(auto it = arr.begin();it!=arr.end();++it)

        if(it->score==max)

            it->show();

}

bool checkString(string s1,string s2){

    int M = s1.length();

    int N = s2.length();

    for (int i = 0; i <= N - M; i++) {

        int j;

        for (j = 0; j < M; j++)

            if (s2[i + j] != s1[j])

                break;

        if (j == M)

            return true;

    }

    return false;

}

void findString(vector<Student> arr,string s){

    for(auto i = arr.begin();i!= arr.end();++i){

        if(checkString(s,i->name))

        i->show();

    }

}

void notString(vector<Student> arr,string s){

    for(auto i = arr.begin();i!= arr.end();++i){

        if(!checkString(s,i->name))

        i->show();

    }

}

int main(){

    vector<Student> arr;

    for(;;){

        cout<<"Enter : \n1. To add a student\n2. To access student by index \n3. To find student with max score \n4. To find number of students\n5. To find student with a substring present in name\n6. To find student without substring present in name\n7. To remove last added student\n8. To show all students\n9. To exit\n";

        int c;

        cin>>c;

        switch(c){

            case 1:{

                Student temp;

                arr.push\_back(temp);

                break;

            }

            case 2:{

                cout<<"Enter index :\n";

                int index;

                cin>>index;

                arr[index].show();

                break;

            }

            case 3:{

                highScore(arr);

                break;

            }

            case 4:{

                cout<<"The number of students is : "<<arr.size()<<endl;

                break;

            }

            case 5:{

                string x;

                cout<<"Enter string to check:\n";

                cin>>x;

                cout<<"The students are:\n";

                findString(arr,x);

                break;

            }

            case 6:{

                string x;

                cout<<"Enter string to check:\n";

                cin>>x;

                cout<<"The students are:\n";

                notString(arr,x);

                break;

            }

            case 7:{

                arr.pop\_back();

                cout<<"Removed\n";

                break;

            }

            case 8:{

                for(auto it = arr.begin();it!=arr.end();++it)

                it->show();

                break;

            }

            case 9:{

                cout<<"Exiting...\n";

                exit(0);

                break;

            }

            default:{

                break;

            }

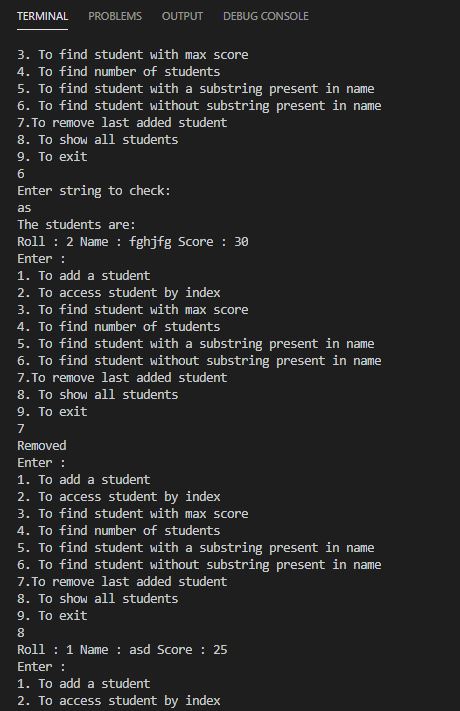
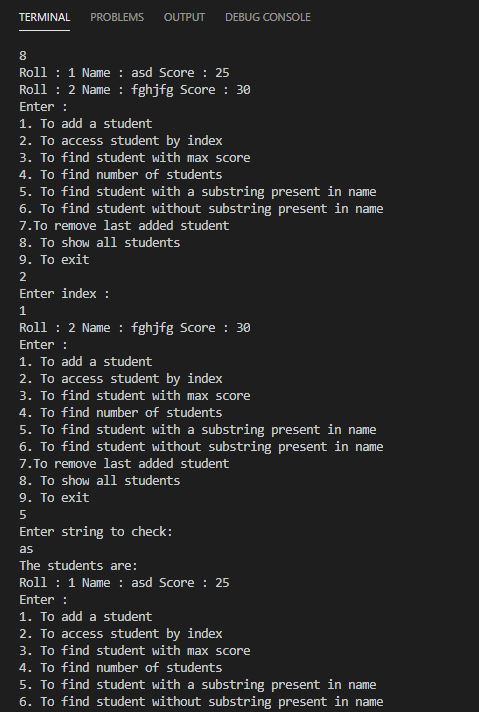
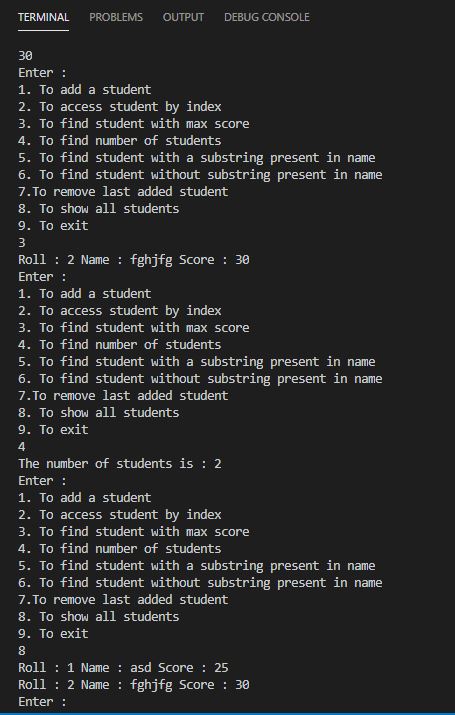
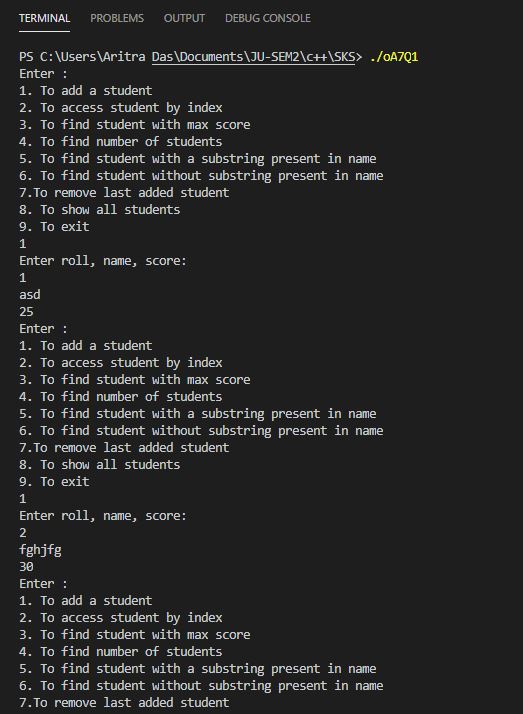
        }

    }

    return 0;

}

Output:



2.

// Create an array of student objects where along with the support mentioned in

// Q.1, one can remove an object with specific roll, sort the collection in the

// descending order and show the same; two student collections can also be

// combined. Take the help of suitable STL class.

#include<iostream>

#include<string>

#include<vector>

#include<algorithm>

using namespace std;

class Student{

    public:

    int roll,score;

    string name;

    Student(){

        cout<<"Enter roll, name, score:\n";

        cin>>roll>>name>>score;

    }

    void show(){

        cout<<"Roll : "<<roll<<" Name : "<<name<<" Score : "<<score<<endl;

    }

};

void highScore(vector<Student> arr){

    int max=-10000;

    for(auto it = arr.begin();it!=arr.end();++it)

        if(it->score>max)

            max = it->score;

    for(auto it = arr.begin();it!=arr.end();++it)

        if(it->score==max)

            it->show();

}

bool checkString(string s1,string s2){

    int M = s1.length();

    int N = s2.length();

    for (int i = 0; i <= N - M; i++) {

        int j;

        for (j = 0; j < M; j++)

            if (s2[i + j] != s1[j])

                break;

        if (j == M)

            return true;

    }

    return false;

}

void findString(vector<Student> arr,string s){

    for(auto i = arr.begin();i!= arr.end();++i){

        if(checkString(s,i->name))

        i->show();

    }

}

void notString(vector<Student> arr,string s){

    for(auto i = arr.begin();i!= arr.end();++i){

        if(!checkString(s,i->name))

        i->show();

    }

}

void removeEl(vector<Student> arr,int x){

    for(auto i = arr.begin();i!= arr.end();++i){

        if(i->roll == x){

            arr.erase(i);

            return;

        }

    }

    cout<<"Roll not found\n";

}

bool compareEl(Student a,Student b){

    if(a.score != b.score)

    return (a.score > b.score);

}

void sortEl(vector<Student> arr){

    sort(arr.begin(),arr.end(),compareEl);

    cout<<"Sorted\n";

    for(auto it = arr.begin();it!=arr.end();++it)

        it->show();

}

int main(){

    vector<Student> arr;

    for(;;){

        cout<<"Enter : \n1. To add a student\n2. To access student by index \n3. To find student with max score \n4. To find number of students\n5. To find student with a substring present in name\n6. To find student without substring present in name\n7. To remove last added student\n8. To show all students\n9. To remove a student\n10. To sort in descending order of score\n11. To exit\n";

        int c;

        cin>>c;

        switch(c){

            case 1:{

                Student temp;

                arr.push\_back(temp);

                break;

            }

            case 2:{

                cout<<"Enter index :\n";

                int index;

                cin>>index;

                arr[index].show();

                break;

            }

            case 3:{

                highScore(arr);

                break;

            }

            case 4:{

                cout<<"The number of students is : "<<arr.size()<<endl;

                break;

            }

            case 5:{

                string x;

                cout<<"Enter string to check:\n";

                cin>>x;

                cout<<"The students are:\n";

                findString(arr,x);

                break;

            }

            case 6:{

                string x;

                cout<<"Enter string to check:\n";

                cin>>x;

                cout<<"The students are:\n";

                notString(arr,x);

                break;

            }

            case 7:{

                arr.pop\_back();

                cout<<"Removed\n";

                break;

            }

            case 8:{

                for(auto it = arr.begin();it!=arr.end();++it)

                it->show();

                break;

            }

            case 9:{

                cout<<"Enter Roll to remove:\n";

                int x;

                cin>>x;

                removeEl(arr,x);

                break;

            }

            case 10:{

                sortEl(arr);

                break;

            }

            case 11:{

                cout<<"Exiting...\n";

                exit(0);

                break;

            }

            default:{

                cout<<"Wrong input\n";

                break;

            }

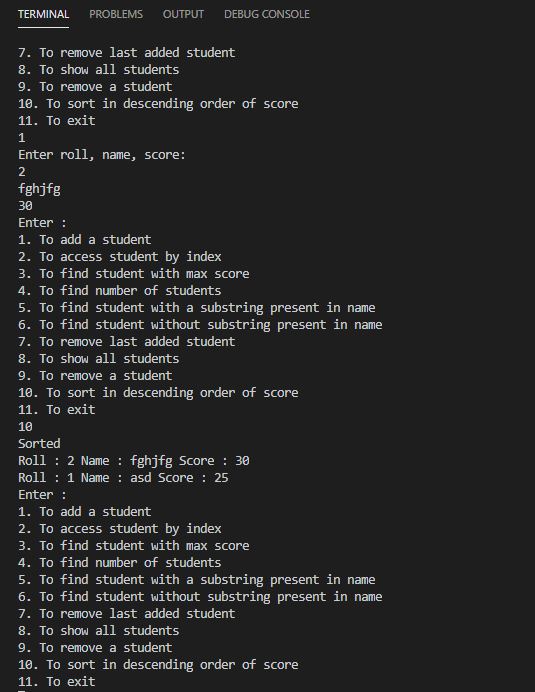
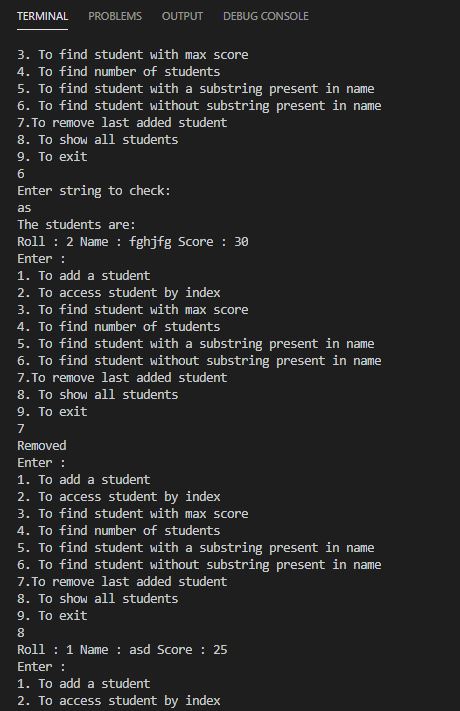
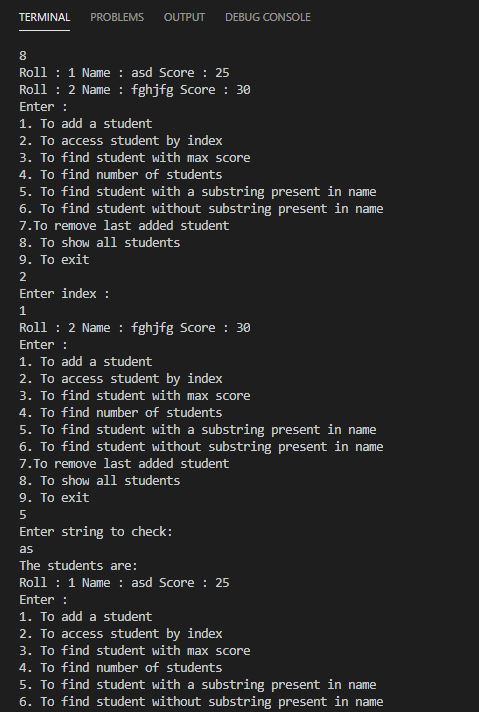
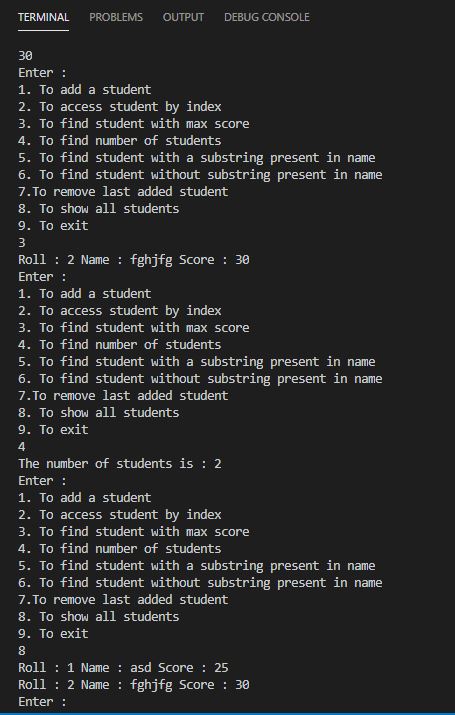
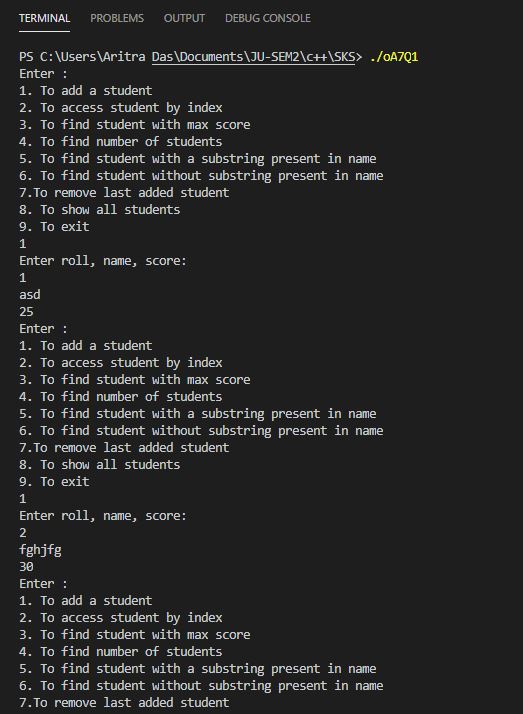
        }

    }

    return 0;

}

Output:



3.

//Students come to mark sheet collection desk and are served in first come first

//served basis. Implement the scenario. Take the help of suitable STL class.

#include<iostream>

#include<string>

#include<queue>

using namespace std;

class Student{

    public:

    int roll,score;

    string name;

    bool marksheet;

    Student(){

        cout<<"Enter roll, name, score:\n";

        cin>>roll>>name>>score;

        marksheet = true;

    }

    void show(){

        cout<<"| Roll : "<<roll<<" Name : "<<name<<" Score : "<<score;

        marksheet?cout<<" Marksheet Issued | ":cout<<"Marksheet not issued | ";

    }

};

void showQ(queue<Student> gq)

{

    queue<Student> g = gq;

    while (!g.empty()) {

        g.front().show();

        g.pop();

        cout<<"     ->      ";

    }

    // cout << '\n';

}

int main(){

    queue<Student> arr;

    for(;;){

        cout<<"Enter : \n1. For entering a student to the queue\n2. For showing queue\n3. To Exit\n";

        int c;

        cin>>c;

        switch(c){

            case 1:{

                Student temp;

                arr.push(temp);

                break;

            }

            case 2:{

                showQ(arr);

                break;

            }

            case 3:{

                cout<<"Exiting...";

                exit(0);

                break;

            }

            default:{

                cout<<"Wrong Input\n";

                break;

            }

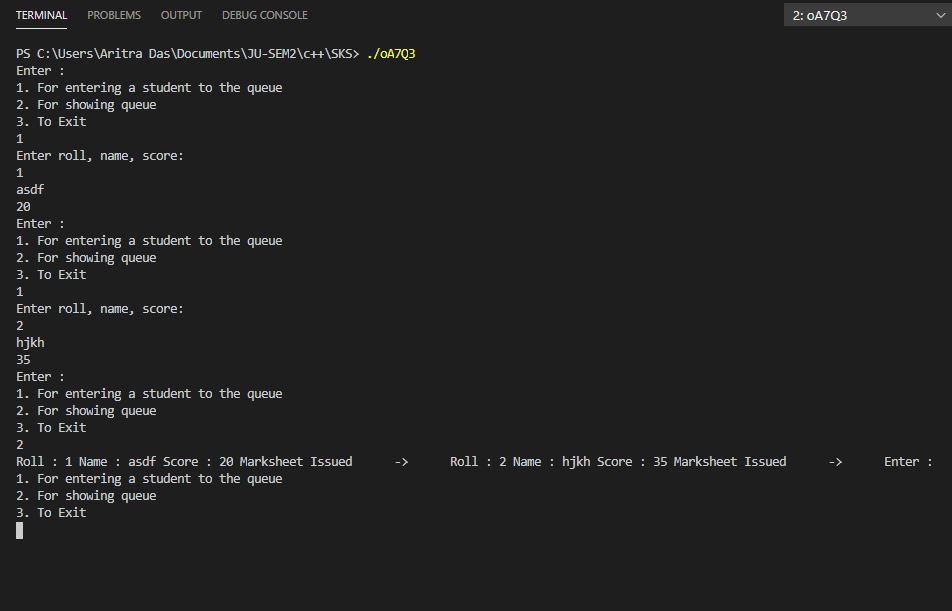
        }

    }

    return 0;

}

Output:



4.

// Maintain a container of students where they are kept in the descending order of

// their scores. Take the help of suitable STL class.

#include<iostream>

#include<string>

#include<set>

#include<algorithm>

#include<iterator>

using namespace std;

class Student{

    public:

    int roll,score;

    string name;

    Student(){

        cout<<"Enter roll, name, score:\n";

        cin>>roll>>name>>score;

    }

    bool operator<(const Student& t) const

    {

        return (this->score > t.score);

    }

};

void showQ(set<Student> arr)

{

    for(auto i = arr.begin();i != arr.end();++i)

    cout<<"| Roll : "<<i->roll<<" Name : "<<i->name<<" Score : "<<i->score<<endl;

}

int main(){

    set<Student> arr;

    for(;;){

        cout<<"Enter : \n1. For entering a student\n2. For showing students\n3. To Exit\n";

        int c;

        cin>>c;

        switch(c){

            case 1:{

                Student temp;

                arr.insert(temp);

                break;

            }

            case 2:{

                showQ(arr);

                break;

            }

            case 3:{

                cout<<"Exiting...";

                exit(0);

                break;

            }

            default:{

                cout<<"Wrong Input\n";

                break;

            }

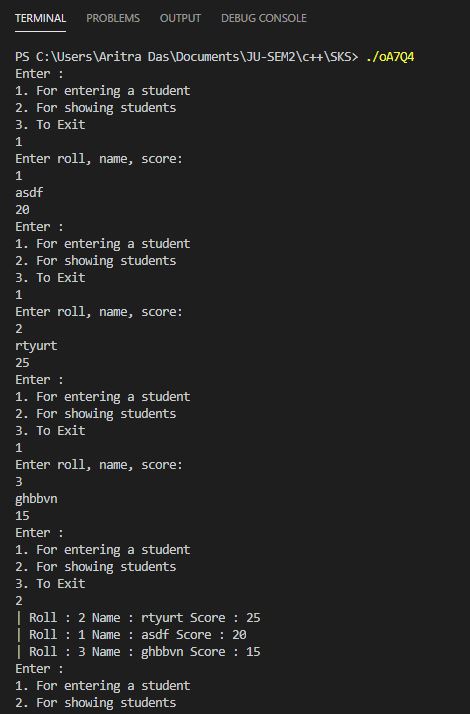
        }

    }

    return 0;

}

Output:



5.

// Store the roll and score of the students in a map in the sorted order of roll. One

// should be able to retrieve the score for a given roll. Take the help of suitable STL

// class.

#include<iostream>

#include<map>

#include<algorithm>

#include<iterator>

using namespace std;

int main(){

    map <int,int> arr;

    for(;;){

        cout<<"Enter :\n1. To insert element to map\n2. To view the whole map\n3. To retrieve particular roll number\n4. To exit\n";

        int c;

        cin>>c;

        switch(c){

            case 1:{

                pair<int,int> temp;

                cout<<"Enter Roll and Score: \n";

                cin>>temp.first>>temp.second;

                arr.insert(temp);

                break;

            }

            case 2:{

                for(auto i = arr.begin();i!=arr.end();++i)

                cout<<"Roll : "<<i->first<<" Score : "<<i->second<<endl;

                break;

            }

            case 3:{

                int x;

                cout<<"Enter roll to find\n";

                cin>>x;

                auto i = arr.find(x);

                if(i!=arr.end())

                cout<<"Roll : "<<i->first<<" Score : "<<i->second<<endl;

                else

                cout<<"Not found\n";

                break;

            }

            case 4:{

                cout<<"Exiting...";

                exit(0);

                break;

            }

            default:{

                break;

            }

        }

    }

    return 0;

}

Output:

