**TASK 01**

#include <iostream>

using namespace std;

void swap(int &num01, int &num02){

    int temp=num01;

    num01=num02;

    num02=temp;

}

template<size\_t size>

void display(int (&array)[size]){

    cout<<endl<<"[ ";

    for(int num : array){

        cout<<num<<", ";

    }

    cout<<"]"<<endl;

}

template<size\_t size>

void bubble\_sort(int (&array)[size]){

    for(size\_t i=0 ; i<size ; i++){

        for(size\_t j=1 ; j<size ; j++){

            if(array[j-1]<array[j]){

                swap(array[j-1],array[j]);

            }

        }

    }

}

int main(){

    int array[10]= {5,1,3,6,2,9,7,4,8,10};

    display(array);

    bubble\_sort(array);

    display(array);

    return 0;

}

**A screenshot of a computer

Description automatically generated**

**TASK 02**

#include <iostream>

#include <string>

using namespace std;

void swap(int &num01, int &num02){

    int temp=num01;

    num01=num02;

    num02=temp;

}

template<size\_t size>

void display(int (&array)[size]){

    cout<<endl<<"[ ";

    for(int num : array){

        cout<<num<<", ";

    }

    cout<<"]"<<endl;

}

int join\_date(int year,int month,int date){

    return ((year\*10000)+(month\*100))+date;

}

int string\_to\_int(string str\_num){

    int int\_num=0;

    for (char character : str\_num) {

        int\_num=int\_num\*10+(character-'0');

    }

    return int\_num;

}

class date{

    private:

        string year;

        string month;

        string day;

        int joined\_date;

    public:

        date():year(""),month(""),day(""),joined\_date(0){};

        date(string \_year, string \_month, string \_day):year(\_year),month(\_month),day(\_day),joined\_date(join\_date(string\_to\_int(\_year),string\_to\_int(\_month),string\_to\_int(\_day))){};

        date(string \_year, string \_month, string \_day,int \_joined\_date):year(\_year),month(\_month),day(\_day),joined\_date(\_joined\_date){};

        void set\_year(string year){

            this->year=year;

        }

        void set\_month(string month){

            this->month=month;

        }

        void set\_day(string day){

            this->day=day;

        }

        void set\_joined\_date(void){

            this->joined\_date=join\_date(string\_to\_int(year),string\_to\_int(month),string\_to\_int(day));

        }

        string get\_year(void){

            return year;

        }

        string get\_month(void){

            return month;

        }

        string get\_day(void){

            return day;

        }

        int get\_joined\_date(void){

            return joined\_date;

        }

        void get\_user\_intput(void){

            cout<<"Input Year: ";

            cin>>year;

            cout<<"Input Month: ";

            cin>>month;

            cout<<"Input day: ";

            cin>>day;

        }

        string get\_date(void){

            string comb\_date=day+"/"+month+"/"+year;

            return comb\_date;

        }

        date& operator = (date obj){

            this->year=obj.year;

            this->month=obj.month;

            this->day=obj.day;

            this->joined\_date=obj.joined\_date;

            return \*this;

        }

};

void swap\_date(date &date01, date &date02){

    date temp=date01;

    date01=date02;

    date02=temp;

}

template<size\_t size>

void selection\_sort(int (&array)[size]){

    for(size\_t i=0 ; i<size-1 ; i++){

        int min=i;

        for(size\_t j=i+1 ; j<size ; j++){

            if(array[min]>array[j]){

                min=j;

            }

        }

        swap(array[i],array[min]);

    }

}

template<size\_t size>

void selection\_sort\_dates(date (&dates)[size]){

    for(size\_t i=0 ; i<size-1 ; i++){

        int min=i;

        for(size\_t j=i+1 ; j<size ; j++){

            if(dates[min].get\_joined\_date()>dates[j].get\_joined\_date()){

                min=j;

            }

        }

        swap\_date(dates[i],dates[min]);

    }

}

int main(){

    date dates[5];

    for(date& date : dates){

        date.get\_user\_intput();

        date.set\_joined\_date();

    }

    system("cls");

    for(date& date : dates){

        cout<<date.get\_date()<<endl;

    }

    selection\_sort\_dates(dates);

    cout<<endl;

    for(date& date : dates){

        cout<<date.get\_date()<<endl;

    }

    return 0;

}

**A computer screen shot of a black screen

Description automatically generated**

**TASK 03**

#include <iostream>

using namespace std;

#define CEO 0

#define CTO 1

#define CFO 2

#define VP 3

#define MGR 4

#define EMP 5

void swap\_desk(int &desk01, int &desk02){

    int temp=desk01;

    desk01=desk02;

    desk02=temp;

}

template<size\_t size>

void display(int (&desks)[size]){

    cout<<endl;

    string designations[6] = {"CEO(Chief Executive Officer)","CTO(Chief Technology Officer)","CFO(Chief Financial Officer)","VP(Vice President)","MGR(Manager)","EMP(Employee)"};

    for(int desk : desks){

        cout<<designations[desk]<<endl;

    }

    cout<<endl;

}

template<size\_t size>

void insertion\_sort(int (&desks)[size]){

    for(size\_t j=1 ; j<size ; j++){

        int key=desks[j];

        int i=j-1;

        while(i>-1 && desks[i]>key){

            desks[i+1]=desks[i];

            i--;

        }

        desks[i+1]=key;

    }

}

int main(){

    int desks[]= {EMP,CFO,MGR,EMP,VP,CTO,MGR,CEO};

    display(desks);

    insertion\_sort(desks);

    cout<<"Desks from left to right:";

    display(desks);

    return 0;

}

**A screenshot of a computer

Description automatically generated**

**TASK 04**

#include <iostream>

using namespace std;

class Employee {

private:

    string name;

    string designation;

    int age;

    float weight;

public:

    Employee() : name(""), designation(""), age(0), weight(0.0) {}

    Employee(string \_name,string \_designation, int \_age, float \_weight)

        : name(\_name), designation(\_designation), age(\_age), weight(\_weight) {}

    string get\_name(){

        return name;

    }

    void set\_name(string name){

        this->name=name;

    }

    string get\_designation(){

        return designation;

    }

    void set\_designation(string designation){

        this->designation=designation;

    }

    int get\_age(){

        return age;

    }

    void set\_age(int age) {

        this->age=age;

    }

    float get\_weight(){

        return weight;

    }

    void set\_weight(float weight) {

        this->weight=weight;

    }

    void display() const {

        cout<<"Name: "<<name<<endl;

        cout<<"Designation: "<<designation<<endl;

        cout<<"Age: "<<age<<endl;

        cout<<"Weight: "<<weight<<" kg"<<endl;

    }

    Employee& operator = (Employee obj){

        this->name=obj.name;

        this->age=obj.age;

        this->designation=obj.designation;

        this->weight=obj.weight;

        return \*this;

    }

};

void swap(int &num01, int &num02){

    int temp=num01;

    num01=num02;

    num02=temp;

}

void reverse(int\* &arr,int size){

    for(int i=0 ; i<size/2 ; i++){

        swap(arr[i],arr[size-1-i]);

    }

}

int\* input\_value(int\* array, int& size, int value) {

    int\* temp=new int[size+1];

    for (int i=0; i<size; ++i) {

        temp[i]=array[i];

    }

    temp[size]=value;

    size++;

    delete[] array;

    return temp;

}

template<size\_t size>

void shell\_sort(Employee (&employees)[size]){

    int k=1, gap\_count=0, gap=1;

    int\* gaps=new int[0];

    while(gap<size){

        gap=(1<<k)-1;

        if(gap<size) gaps=input\_value(gaps,gap\_count,gap);

        k++;

    }

    reverse(gaps,gap\_count);

    for (int i=0; i<gap\_count; i++) {

        int gap=gaps[i];

        for (int j=gap; j<size; j++) {

            Employee key=employees[j];

            int res=j;

            while (res>=gap && employees[res-gap].get\_weight()>key.get\_weight()){

                employees[res]=employees[res-gap];

                res-=gap;

            }

            employees[res]=key;

        }

    }

    delete[] gaps;

}

int main(){

    Employee employees[10] = {

        Employee("Alice", "Manager", 34, 72.0),

        Employee("Bob", "Developer", 29, 65.5),

        Employee("Charlie", "Designer", 31, 68.0),

        Employee("Diana", "Analyst", 27, 70.5),

        Employee("Edward", "Tester", 40, 78.2),

        Employee("Fiona", "Developer", 33, 60.1),

        Employee("George", "Manager", 45, 80.4),

        Employee("Hannah", "Designer", 26, 62.5),

        Employee("Ian", "Analyst", 39, 76.0),

        Employee("Jenna", "Tester", 24, 55.8),

    };

    cout<<"Unsorted Employees"<<endl;

    for(int i=0; i<10; ++i) {

        cout << "Employee " << (i + 1) << ":" << endl;

        employees[i].display();

        cout << endl;

        // cout<<employees[i].get\_weight()<<" ";

    }

    cout<<endl<<endl;

    shell\_sort(employees);

    cout<<endl<<endl<<"Sorted Employees"<<endl;

    for(int i=0; i<10; ++i) {

        cout << "Employee " << (i + 1) << ":" << endl;

        employees[i].display();

        cout << endl;

        // cout<<employees[i].get\_weight()<<" ";

    }

    return 0;

}

**A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated**

**TASK 05**

#include <iostream>

using namespace std;

void swap(float &num01, float &num02){

    int temp=num01;

    num01=num02;

    num02=temp;

}

template<size\_t size>

void display(float (&array)[size]){

    cout<<endl<<"[ ";

    for(float num : array){

        cout<<num<<", ";

    }

    cout<<"]"<<endl;

}

template<size\_t size>

void comb\_sort(float (&list)[size]){

    float shrink=10;

    int gap=size;

    bool swapped=true;

    while(gap>1 || swapped){

        gap = int(gap/shrink);

        if(gap<1) gap=1;

        swapped=false;

        for(int i=0 ; i+gap<size ; i++){

            if(list[i]>list[i+gap]){

                swap(list[i],list[i+gap]);

                swapped=true;

            }

        }

    }

}

int main(){

    float price\_list[10]= {5.2,112.3,33.53,60.9,210.33,93.4,741.,444.4,853.3,10.01};

    display(price\_list);

    comb\_sort(price\_list);

    display(price\_list);

    return 0;

}

**A screenshot of a computer

Description automatically generated**

**TASK 06**

#include <iostream>

using namespace std;

void display(int\* &array,int size){

    cout<<endl<<"[ ";

    for(int i=0 ; i<size ; i++){

        cout<<array[i]<<", ";

    }

    cout<<"]"<<endl;

}

void swap(int &num01, int &num02){

    int temp=num01;

    num01=num02;

    num02=temp;

}

void sort(int\* &array,int size){

    for(size\_t i=0 ; i<size ; i++){

        for(size\_t j=i ; j<size ; j++){

            if(array[i]>array[j]){

                swap(array[i],array[j]);

            }

        }

    }

}

int binary\_search(int\* &arr,int target,int left,int right){

    if(left>right) return (-1\*left);

    int mid=(left+right)/2;

    if(arr[mid]==target) return mid;

    else if(arr[mid]<target) return binary\_search(arr, target, mid+1, right);

    else return binary\_search(arr, target, left, mid-1);

}

int\* input\_value(int\* array, int& size,int index, int value) {

    int\* temp=new int[size+1];

    int j=0;

    for (int i=0; i<size; ++i) {

        if(i!=index){

            temp[i]=array[j]; j++;

        }else{

            temp[i]=value;

        }

    }

    temp[size]=value;

    size++;

    delete[] array;

    return temp;

}

int main(){

    // ID: 23K-0647 -> 47

    int size=20;

    int\* id = new int[size];

    int random\_ids[]={22,47,56,87,90,54,14,54,34,65,76,11,77,42,67,88,53,23,54,23};

    for (int i = 0; i < 20; i++) {

        id[i] = random\_ids[i];

    }

    sort(id,size);

    display(id,size);

    cout<<endl;

    int index=binary\_search(id,47,0,size-1);

    if(index<0){

        id=input\_value(id,size,abs(index),47);

        cout<<"Id not found"<<endl<<"Id added to the list at index: "<<abs(index)-1<<endl;

    }else{

        cout<<"Id found in the list at index: "<<index<<endl;

    }

    display(id,size);

    return 0;

}

**If 47 exits**

**A number on a black background

Description automatically generated**

**If it does not**

**A black background with white numbers

Description automatically generated**

**TASK 07**

#include <iostream>

using namespace std;

void display(int\* &array,int size){

    cout<<endl<<"[ ";

    for(int i=0 ; i<size ; i++){

        cout<<array[i]<<", ";

    }

    cout<<"]"<<endl;

}

void swap(int &num01, int &num02){

    int temp=num01;

    num01=num02;

    num02=temp;

}

void sort(int\* &array,int size){

    for(size\_t i=0 ; i<size ; i++){

        for(size\_t j=i ; j<size ; j++){

            if(array[i]>array[j]){

                swap(array[i],array[j]);

            }

        }

    }

}

int interpolation\_search(int\* &arr,int target,int left,int right){

    if(left>right) return -1;

    int mdataset=left+(((right-left)/(arr[right]-arr[left])))\*(target-arr[left]);

    if(arr[mdataset]==target) return mdataset;

    else if(arr[mdataset]<target) return interpolation\_search(arr, target, mdataset+1, right);

    else return interpolation\_search(arr, target, left, mdataset-1);

}

int\* input\_value(int\* array, int& size, int value) {

    int\* temp=new int[size+1];

    for(int i=0; i<size; ++i) {

        temp[i]=array[i];

    }

    temp[size]=value;

    size++;

    delete[] array;

    return temp;

}

bool check\_uniformity(int\* dataset,int size){

    int diff = dataset[1]-dataset[0];

    for (int i=2; i<size; i++) {

        if (dataset[i]-dataset[i-1]!=diff) return false;

    }

    return true;

}

int main(){

    int size=0;

    int\* dataset = new int[size];

    bool stop=false;

    int value;

    char choice;

    while(!stop){

        cout<<"Input Data Element:";

        cin>>value;

        dataset=input\_value(dataset,size,value);

        cout<<"Do You Want To Stop (y/n): ";

        cin>>choice;

        if(choice=='y') stop=true;

    }

    system("cls");

    if(!check\_uniformity(dataset,size)) throw("Ununiform data");

    sort(dataset,size);

    int target;

    cout<<"Input Target value: ";

    cin>>target;

    int index=interpolation\_search(dataset,target,0,size-1);

    if(index<0){

        cout<<"Index & Value not found";

    } else {

        cout<<"Target Value At Index: "<<index;

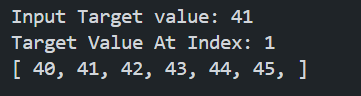
    }

    display(dataset,size);

    cout<<endl;

    return 0;

}

****