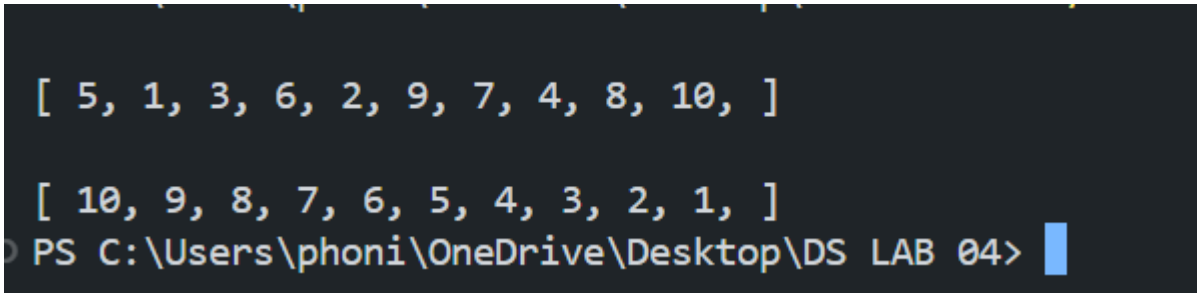
TASK 01

```cpp
#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;
}
```



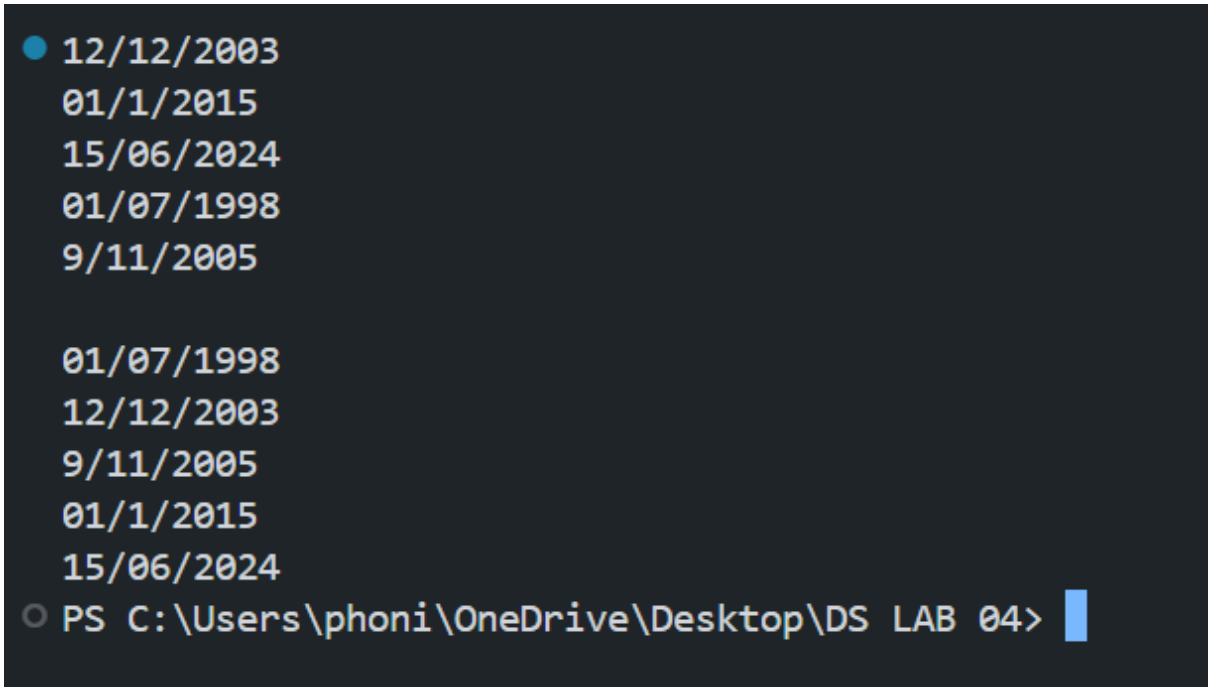
TASK 02

```cpp
#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;
}
```


12/12/2003
01/1/2015
15/06/2024
01/07/1998
9/11/2005
01/07/1998
12/12/2003
9/11/2005
01/1/2015
15/06/2024
PS C:\Users\phoni\OneDrive\Desktop\DS LAB 04>

TASK 03

```cpp
#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;
}
```

```
EMP(Employee)
CFO(Chief Financial Officer)
MGR(Manager)
EMP(Employee)
VP(Vice President)
CTO(Chief Technology Officer)
MGR(Manager)
CEO(Chief Executive Officer)

Desks from left to right:
CEO(Chief Executive Officer)
CTO(Chief Technology Officer)
CFO(Chief Financial Officer)
VP(Vice President)
MGR(Manager)
MGR(Manager)
EMP(Employee)
EMP(Employee)
```

TASK 04

```cpp
#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;
}
```

```cpp
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;
}
```

```
Unsorted Employees
Employee 1:
Name: Alice
Designation: Manager
Age: 34
Weight: 72 kg

Employee 2:
Name: Bob
Designation: Developer
Age: 29
Weight: 65.5 kg

Employee 3:
Name: Charlie
Designation: Designer
Age: 31
Weight: 68 kg

Employee 4:
Name: Diana
Designation: Analyst
Age: 27
Weight: 70.5 kg
```

```
Employee 5:
Name: Edward
Designation: Tester
Age: 40
Weight: 78.2 kg

Employee 6:
Name: Fiona
Designation: Developer
Age: 33
Weight: 60.1 kg

Employee 7:
Name: George
Designation: Manager
Age: 45
Weight: 80.4 kg

Employee 8:
Name: Hannah
Designation: Designer
Age: 26
Weight: 62.5 kg

Employee 9:
Name: Ian
Designation: Analyst
Age: 39
Weight: 76 kg

Employee 10:
Name: Jenna
Designation: Tester
Age: 24
Weight: 55.8 kg
```

```
Sorted Employees
Employee 1:
Name: Jenna
Designation: Tester
Age: 24
Weight: 55.8 kg

Employee 2:
Name: Fiona
Designation: Developer
Age: 33
Weight: 60.1 kg

Employee 3:
Name: Hannah
Designation: Designer
Age: 26
Weight: 62.5 kg

Employee 4:
Name: Bob
Designation: Developer
Age: 29
Weight: 65.5 kg

Employee 5:
Name: Charlie
Designation: Designer
Age: 31
Weight: 68 kg

Employee 6:
Name: Diana
Designation: Analyst
Age: 27
Weight: 70.5 kg
```

```
Employee 7:
Name: Alice
Designation: Manager
Age: 34
Weight: 72 kg

Employee 8:
Name: Ian
Designation: Analyst
Age: 39
Weight: 76 kg

Employee 9:
Name: Edward
Designation: Tester
Age: 40
Weight: 78.2 kg

Employee 10:
Name: George
Designation: Manager
Age: 45
Weight: 80.4 kg

PS C:\Users\phoni\OneDrive\Desktop\DS LAB 04>
```

TASK 05

```cpp
#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;
}
```

```
[ 5.2, 112.3, 33.53, 60.9, 210.33, 93.4, 741, 444.4, 853.3, 10.01, ]

[ 5.2, 10.01, 33, 60, 93, 112, 210, 444, 741, 853, ]
PS C:\Users\phoni\OneDrive\Desktop\DS LAB 04>
```

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

```
[ 11, 14, 22, 23, 23, 34, 42, 47, 53, 54, 54, 54, 56, 65, 67, 76, 77, 87, 88, 90, ]

Id found in the list at index: 7

[ 11, 14, 22, 23, 23, 34, 42, 47, 53, 54, 54, 54, 56, 65, 67, 76, 77, 87, 88, 90, ]
```

If it does not

```
[ 11, 14, 22, 23, 23, 34, 42, 45, 53, 54, 54, 54, 56, 65, 67, 76, 77, 87, 88, 90, ]

Id not found
Id added to the list at index: 7

[ 11, 14, 22, 23, 23, 34, 42, 45, 47, 53, 54, 54, 54, 56, 65, 67, 76, 77, 87, 88, 47, ]
```

TASK 07

```cpp
#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;
}
```

```
Input Target value: 41
Target Value At Index: 1
[ 40, 41, 42, 43, 44, 45, ]
```