

NAME	SAID RASOOL
SAP	55691
SCE	SE 3-2

LAB_13.

TASK.01:

INPUT:

```
#include <iostream>
```

```
using namespace std;
```

```
void bubbleSortDescending(int arr[], int n)
```

```
{
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] < arr[j + 1]) {

                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}
```

```
void printArray(int arr[], int n)
```

```
{
    for (int i = 0; i < n; i++) {
        std::cout << arr[i] << " ";
    }
}
```

```

    }

    std::cout << std::endl;
}

int main()
{
    int arr[] = {44391, 44647, 47777, 53759, 55181, 55223, 55225, 55330, 55349, 55356,
                55405, 55434, 55465, 55469, 55566, 55579, 55584, 55590, 55632, 55633,
                55652, 55691, 55700, 55766, 55780, 55843, 55853};

    int n = sizeof(arr) / sizeof(arr[0]);

    bubbleSortDescending(arr, n);

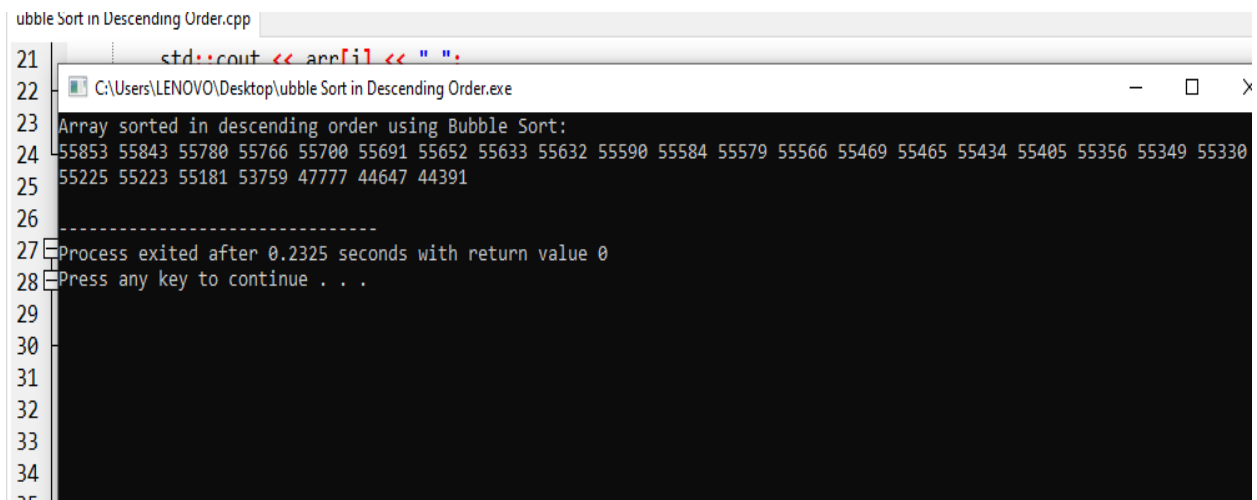
    std::cout << "Array sorted in descending order using Bubble Sort:\n";

    printArray(arr, n);

    return 0;
}

```

OUTPUT:



The screenshot shows a Windows command prompt window titled "C:\Users\LENOVO\Desktop\ubble Sort in Descending Order.exe". The output of the program is as follows:

```

21  std::cout << arr[i] << " ";
22
23  Array sorted in descending order using Bubble Sort:
24  55853 55843 55780 55766 55700 55691 55652 55633 55632 55590 55584 55579 55566 55469 55465 55434 55405 55356 55349 55330
25  55225 55223 55181 53759 47777 44647 44391
26  -----
27  Process exited after 0.2325 seconds with return value 0
28  Press any key to continue . . .
29
30
31
32
33
34

```

TASK.02:

INPUT:

```
#include <iostream>
```

```
void bubbleSortAscending(int arr[], int n) {  
    for (int i = 0; i < n - 1; i++) {  
        for (int j = 0; j < n - i - 1; j++) {  
            if (arr[j] > arr[j + 1]) {  
                // Swap arr[j] and arr[j+1] to arrange in ascending order  
                int temp = arr[j];  
                arr[j] = arr[j + 1];  
                arr[j + 1] = temp;  
            }  
        }  
    }  
}
```

```
void printArray(int arr[], int n) {  
    for (int i = 0; i < n; i++) {  
        std::cout << arr[i] << " ";  
    }  
    std::cout << std::endl;  
}
```

```
int main() {
```

```

int arr[] = {44391, 44647, 47777, 53759, 55181, 55223, 55225, 55330, 55349, 55356,
            55405, 55434, 55465, 55469, 55566, 55579, 55584, 55590, 55632, 55633,
            55652, 55691, 55700, 55766, 55780, 55843, 55853};

int n = sizeof(arr) / sizeof(arr[0]);

bubbleSortAscending(arr, n);

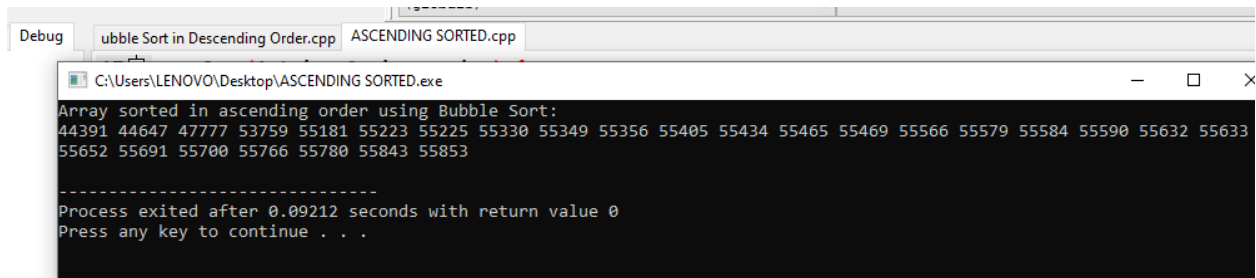
std::cout << "Array sorted in ascending order using Bubble Sort:\n";

printArray(arr, n);

return 0;
}

```

OUTPUT:



The screenshot shows a Windows command prompt window titled "C:\Users\LENOVO\Desktop\ASCENDING SORTED.exe". The output of the program is as follows:

```

Array sorted in ascending order using Bubble Sort:
44391 44647 47777 53759 55181 55223 55225 55330 55349 55356 55405 55434 55465 55469 55566 55579 55584 55590 55632 55633
55652 55691 55700 55766 55780 55843 55853

-----
Process exited after 0.09212 seconds with return value 0
Press any key to continue . . .

```

Task.02:

Input:

```
#include <iostream>
```

```

void insertionSortAscending(int arr[], int n) {

    for (int i = 1; i < n; i++) {

        int key = arr[i];

```

```

    int 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 && arr[j] > key) {
        arr[j + 1] = arr[j];
        j = j - 1;
    }
    arr[j + 1] = key;
}
}

```

```

void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        std::cout << arr[i] << " ";
    }
    std::cout << std::endl;
}

```

```

int main() {
    int arr[] = {44391, 44647, 47777, 53759, 55181, 55223, 55225, 55330, 55349, 55356,
        55405, 55434, 55465, 55469, 55566, 55579, 55584, 55590, 55632, 55633,
        55652, 55691, 55700, 55766, 55780, 55843, 55853};
    int n = sizeof(arr) / sizeof(arr[0]);

    insertionSortAscending(arr, n);
}

```

```

std::cout << "Array sorted in ascending order using Insertion Sort:\n";

printArray(arr, n);

return 0;
}

```

Output:

```

in Descending Order.cpp  ASCENDING SORTED.cpp  insertion ascending.cpp
for (int i = 0; i < n; i++) {
    std::cout << arr[i] << " ";
}

C:\Users\LENOVO\Desktop\insertion ascending.exe
Array sorted in ascending order using Insertion Sort:
4391 44647 47777 53759 55181 55223 55225 55330 55349 55356 55405 55434 55465 55469 55566 55579 55584
5652 55691 55700 55766 55780 55843 55853

-----
Process exited after 0.0865 seconds with return value 0
Press any key to continue . . .

```

Task.04:

Input:

```
#include <iostream>
```

```
void insertionSortDescending(int arr[], int n) {
```

```
    for (int i = 1; i < n; i++) {
```

```
        int key = arr[i];
```

```
        int j = i - 1;
```

```
        while (j >= 0 && arr[j] < key) {
```

```
            arr[j + 1] = arr[j];
```

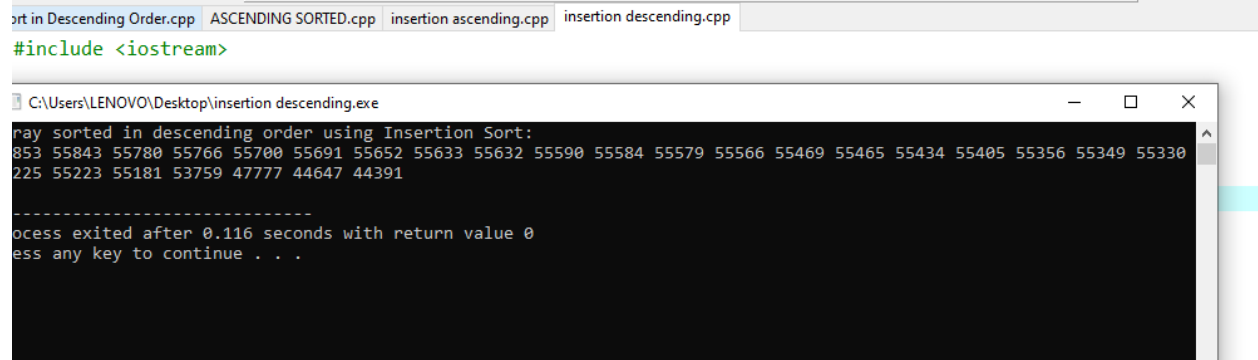
```
            j = j - 1;
```

```
    }  
    arr[j + 1] = key;  
}  
}
```

```
void printArray(int arr[], int n) {  
    for (int i = 0; i < n; i++) {  
        std::cout << arr[i] << " ";  
    }  
    std::cout << std::endl;  
}
```

```
int main()  
{  
    int arr[] = {44391, 44647, 47777, 53759, 55181, 55223, 55225, 55330, 55349, 55356,  
                55405, 55434, 55465, 55469, 55566, 55579, 55584, 55590, 55632, 55633,  
                55652, 55691, 55700, 55766, 55780, 55843, 55853};  
    int n = sizeof(arr) / sizeof(arr[0]);  
  
    insertionSortDescending(arr, n);  
    std::cout << "Array sorted in descending order using Insertion Sort:\n";  
    printArray(arr, n);  
  
    return 0;  
}
```

Output:



```
rt in Descending Order.cpp | ASCENDING SORTED.cpp | insertion ascending.cpp | insertion descending.cpp
#include <iostream>

C:\Users\LENOVO\Desktop\insertion descending.exe
ray sorted in descending order using Insertion Sort:
853 55843 55780 55766 55700 55691 55652 55633 55632 55590 55584 55579 55566 55469 55465 55434 55405 55356 55349 55330
225 55223 55181 53759 47777 44647 44391

-----
rocess exited after 0.116 seconds with return value 0
ess any key to continue . . .
```

Task.05:

Input:

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
class Person {
```

```
private:
```

```
    int per_id;
```

```
    string per_name;
```

```
    int per_age;
```

```
public:
```

```
    Person(int id = 0, string name = "", int age = 0)
```

```
    { per_id(id), per_name(name), per_age(age) {}
```



```

// Input function
void input() {
    cout << "Enter Person ID: ";
    cin >> per_id;
    cout << "Enter Person Name: ";
    cin.ignore();
    getline(cin, per_name);
    cout << "Enter Person Age: ";
    cin >> per_age;
}

void output() const {
    cout << "ID: " << per_id << ", Name: " << per_name << ", Age: " << per_age << endl;
}

};

class LinkedList {
private:
    struct Node {
        Person person;
        Node* next;
        Node(const Person& p) : person(p), next(NULL) {}
    };
};

```

```
Node* head;
```

```
public:
```

```
LinkedList() : head(NULL) {}
```

```
~LinkedList() {
```

```
    while (head) {
```

```
        Node* temp = head;
```

```
        head = head->next;
```

```
        delete temp;
```

```
    }
```

```
}
```

```
void insertAtHead(const Person& p) {
```

```
    Node* newNode = new Node(p);
```

```
    newNode->next = head;
```

```
    head = newNode;
```

```
}
```

```
void display() const {
```

```
    Node* current = head;
```

```
    while (current) {
```

```
        current->person.output();
```

```
        current = current->next;

    }

}

};
```

```
int main() {

    LinkedList list;

    int choice;

    do {

        cout << "\n1. Add Person\n2. Display All Persons\n3. Exit\nEnter your choice: ";

        cin >> choice;

        switch (choice) {

            case 1: {

                Person p;

                p.input();

                list.insertAtHead(p);

                break;

            }

            case 2:

                list.display();

                break;

            case 3:

                cout << "Exiting..." << endl;
```

```

        break;

    default:

        cout << "Invalid choice! Please try again." << endl;

    }

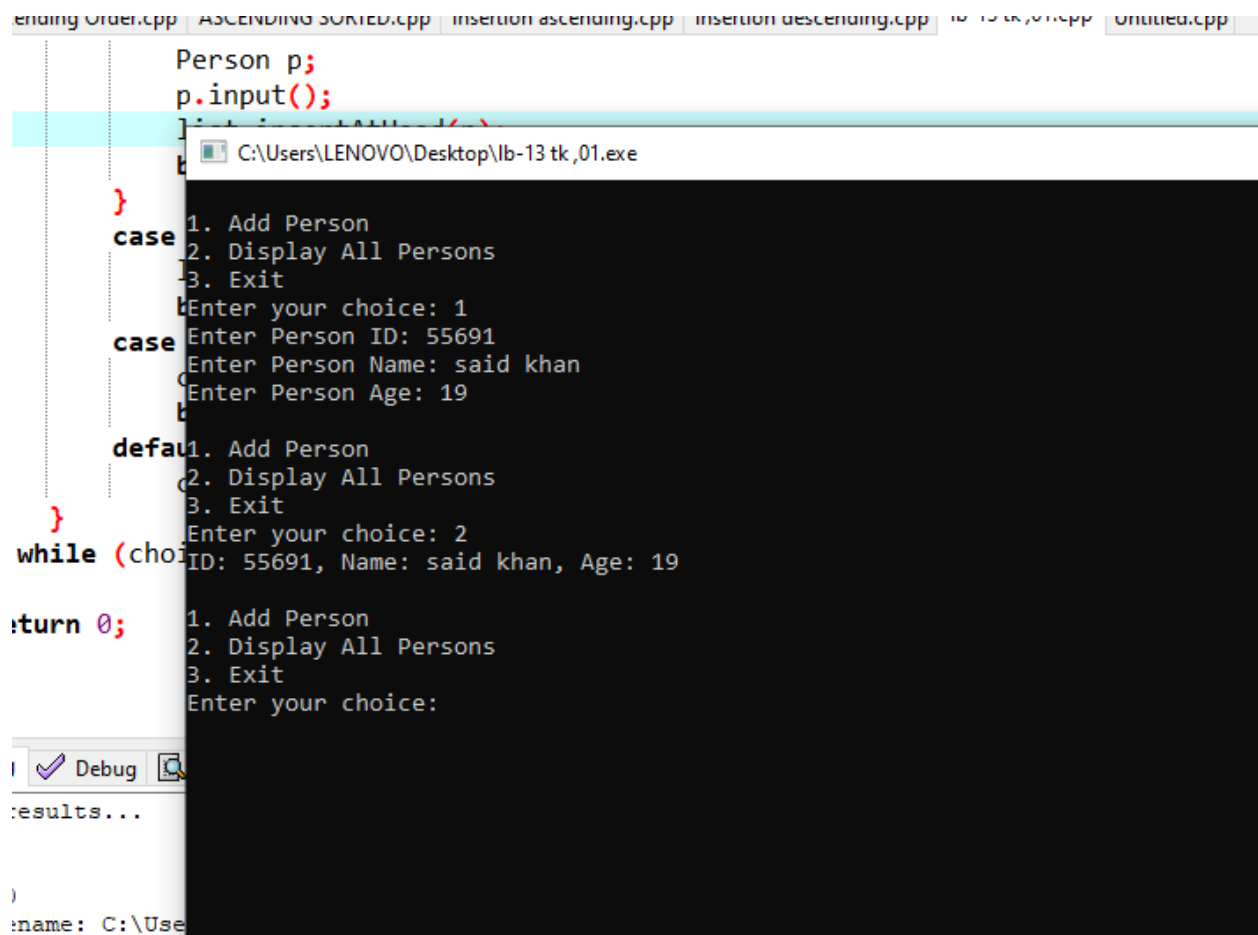
} while (choice != 3);

return 0;

}

```

Output:



The screenshot shows a C++ IDE with a code editor on the left and a console window on the right. The code editor displays a menu-driven program for managing a list of persons. The console window shows the program's execution, including the menu display, user input for choice 1, and the subsequent prompts for ID, name, and age.

```

ending Order.cpp | ASCENDING SORTED.cpp | insertion ascending.cpp | insertion descending.cpp | lb-13 tk,01.cpp |Untitled.cpp
Person p;
p.input();
}
case 1: Add Person
case 2: Display All Persons
case 3: Exit
Enter your choice: 1
Enter Person ID: 55691
Enter Person Name: said khan
Enter Person Age: 19
default:
1. Add Person
2. Display All Persons
3. Exit
Enter your choice: 2
ID: 55691, Name: said khan, Age: 19
while (choice != 3)
return 0;
1. Add Person
2. Display All Persons
3. Exit
Enter your choice:
results...
name: C:\Use

```

Task.06:

Input:

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
class Node {
```

```
public:
```

```
    int data;
```

```
    bool insertedAtHead;
```

```
    Node* next;
```

```
    Node(int value, bool insertedAtHead) {
```

```
        data = value;
```

```
        this->insertedAtHead = insertedAtHead;
```

```
        next = NULL;
```

```
    }
```

```
};
```

```
class LinkedList {
```

```
private:
```

```
    Node* head;
```

```
    Node* tail;
```

```
    bool insertAtHeadNext;
```

public:

```
LinkedList() : head(NULL), tail(NULL), insertAtHeadNext(true) {}
```

```
// Destructor to delete all nodes and free memory
```

```
~LinkedList() {
```

```
    while (head) {
```

```
        Node* temp = head;
```

```
        head = head->next;
```

```
        delete temp;
```

```
    }
```

```
}
```

```
void insert(int value) {
```

```
    if (insertAtHeadNext) {
```

```
        // Insert at head
```

```
        Node* newNode = new Node(value, true);
```

```
        newNode->next = head;
```

```
        head = newNode;
```

```
    } else {
```

```
        // Insert at tail
```

```
        Node* newNode = new Node(value, false);
```

```
        if (tail) {
```

```
            tail->next = newNode;
```

```

        tail = newNode;

    } else {

        head = newNode;
        tail = newNode;

    }

    }

    insertAtHeadNext = !insertAtHeadNext;
}

void display() const {
    Node* current = head;
    while (current) {
        cout << "Data: " << current->data
            << " | Inserted from: " << (current->insertedAtHead ? "Head" : "Tail") << endl;
        current = current->next;
    }
}

};

int main() {
    LinkedList list;

    int choice;

    do {

```

```

cout << "\n1. Insert Node\n2. Display All Nodes\n3. Exit\nEnter your choice: ";

cin >> choice;

switch (choice) {
    case 1: {
        int value;

        cout << "Enter value to insert: ";

        cin >> value;

        list.insert(value);

        break;
    }
    case 2:
        list.display();

        break;
    case 3:
        cout << "Exiting..." << endl;

        break;
    default:
        cout << "Invalid choice! Please try again." << endl;
}
} while (choice != 3);

return 0;
}

```

OUTPUT:

