Comsats University Islamabad, Vehari Campus

Academic Year 2022 -23

Department: Computer Science

Full Name: Sana Ahmad

Roll No.: SP22-BCS-129

Section: "B"

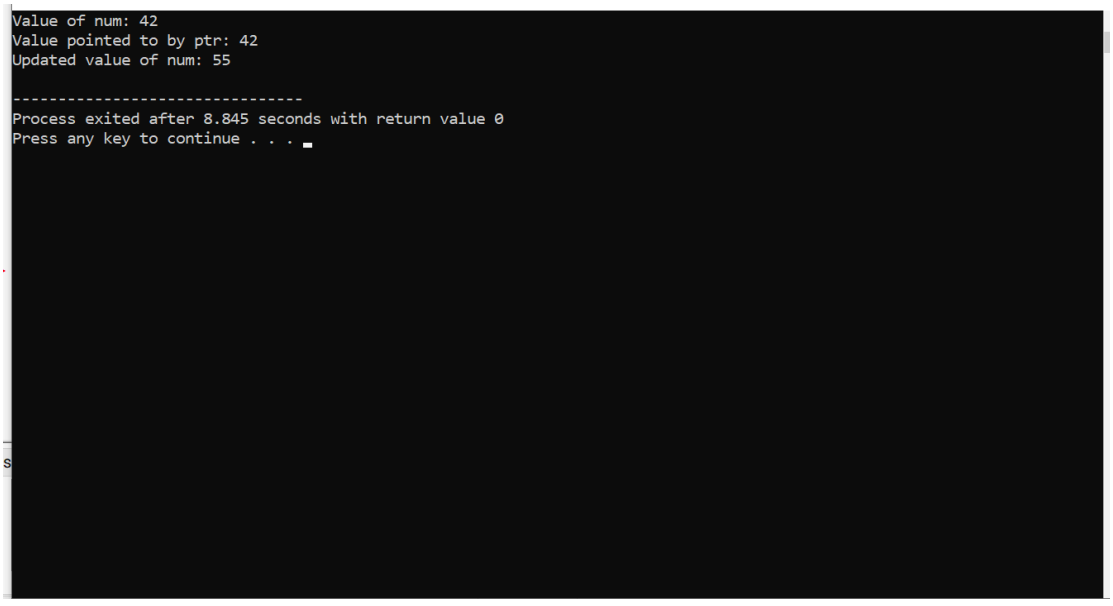
Subject: Data Structure and Algorithm

Date: 11/09/2023

Submitted to: Mam Yasmeen

```
#include <iostream>

int main() {
    int num = 42;     // Declare an integer variable 'num' and assign a value to it
    int* ptr = &num;  // Declare a pointer to an integer and assign the address of 'num' to it

    // Print the value of 'num' and the value pointed to by 'ptr'
    std::cout << "Value of num: " << num << std::endl;
    std::cout << "Value pointed to by ptr: " << *ptr << std::endl;

    // Modify the value of 'num' through the pointer 'ptr'
    *ptr = 55;

    // Print the updated value of 'num'
    std::cout << "Updated value of num: " << num << std::endl;

    return 0;
}
```

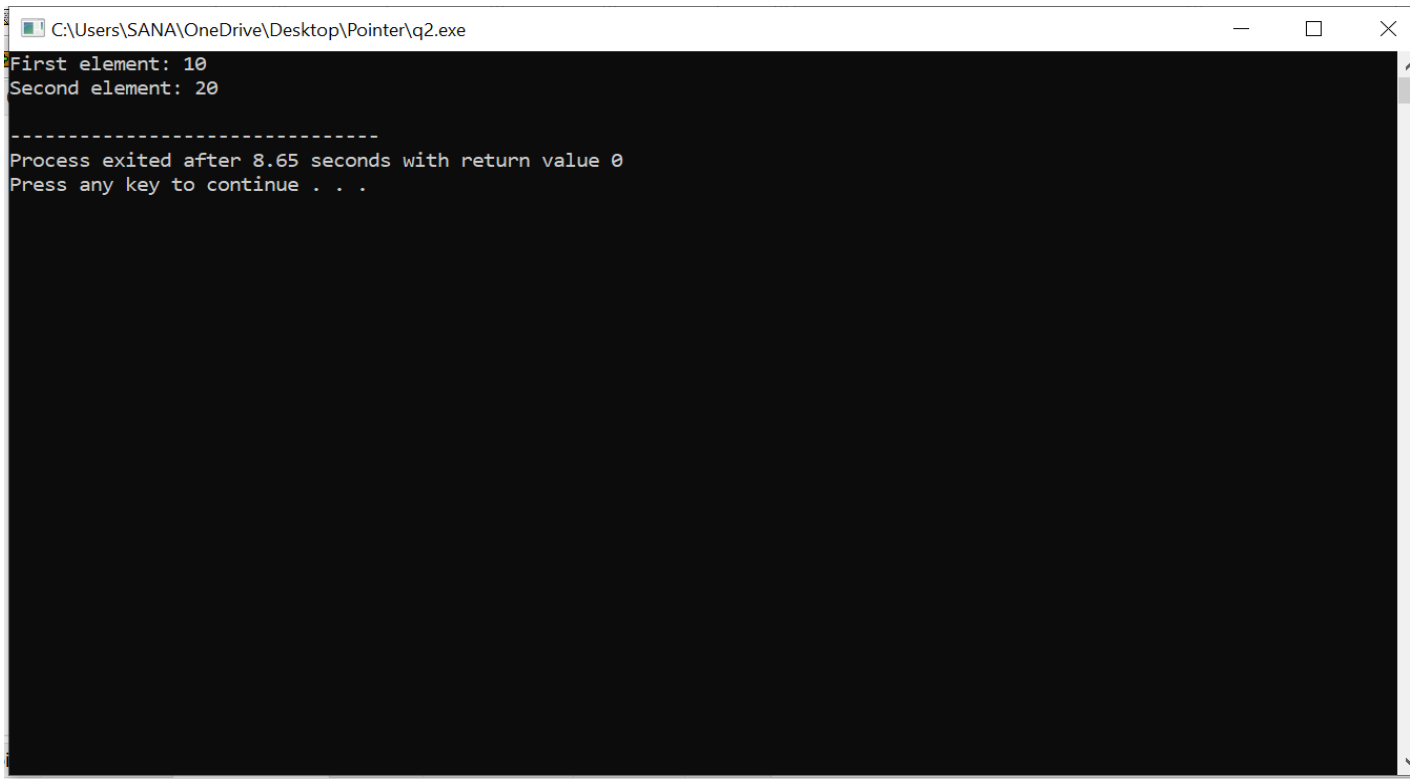
Output:



Problem: 2

```
#include <iostream>


int main() {
    int arr[] = {10, 20, 30, 40};
    int* ptr = arr;
    std::cout << "First element: " << *ptr << std::endl;
    ptr++; // Move to the next element
    std::cout << "Second element: " << *ptr << std::endl;
    return 0;
}
```

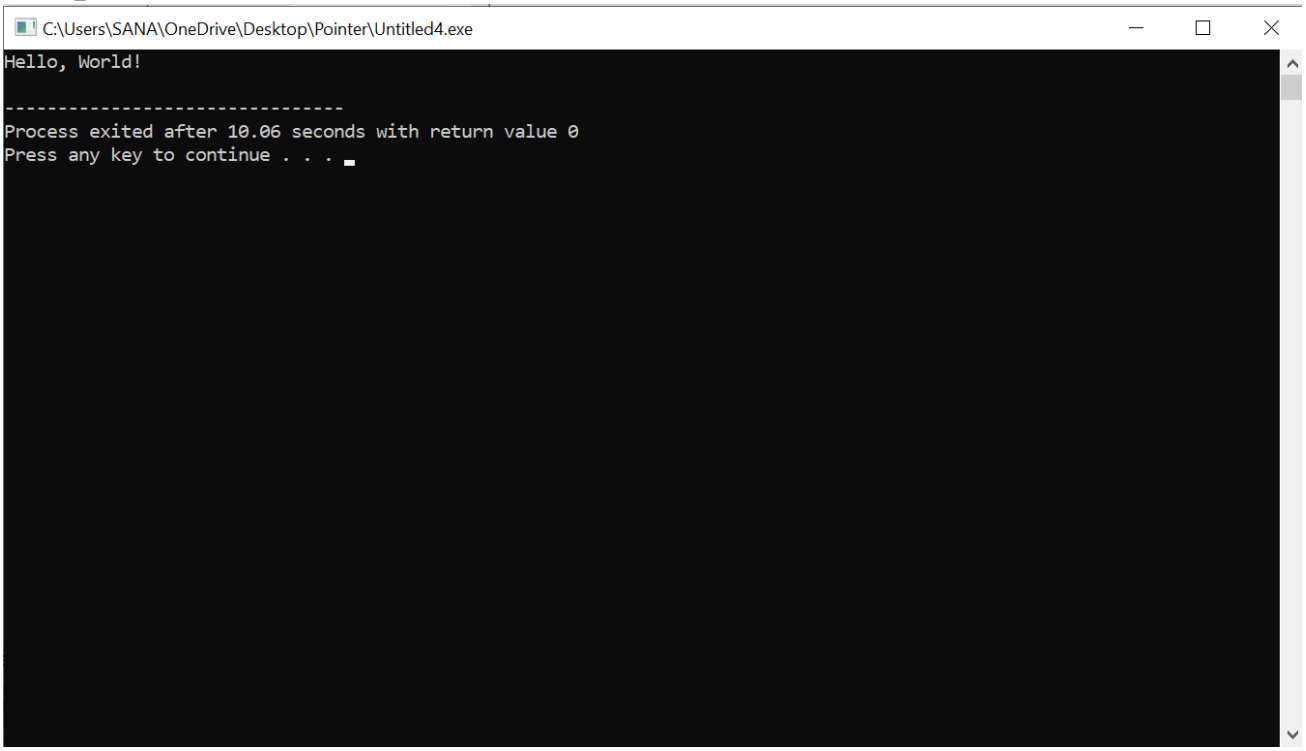
Output:



Problem: 3

```
#include <iostream>


void greet() {
   std::cout << "Hello, World!" << std::endl;
}


int main() {
   void (*functionPtr)() = greet;
   functionPtr(); // Call the function through the pointer
   return 0;
}
```

Output:

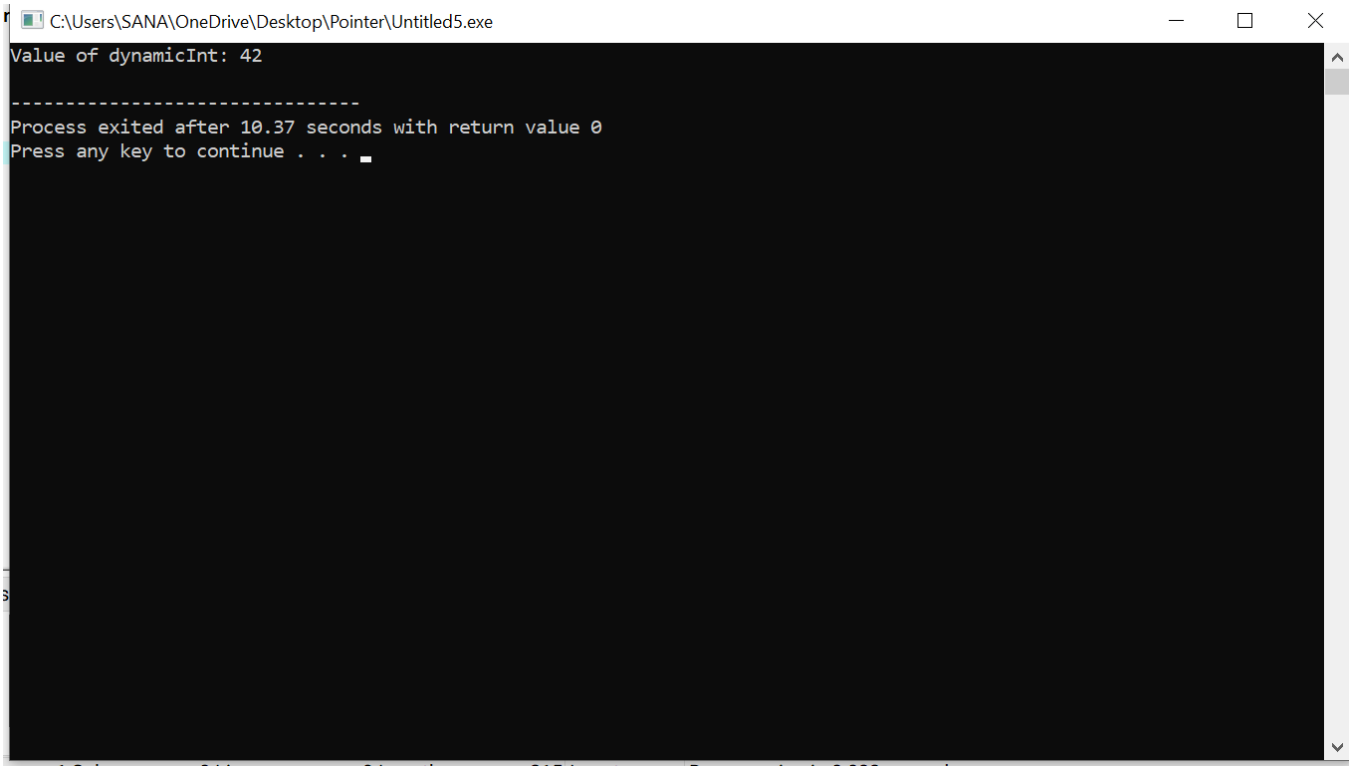


Problem: 4

```
#include <iostream>

int main() {
```

```
    int* dynamicInt = new int(42);
    std::cout << "Value of dynamicInt: " << *dynamicInt << std::endl;
    delete dynamicInt; // Free the allocated memory
    return 0;
}
```

Output:

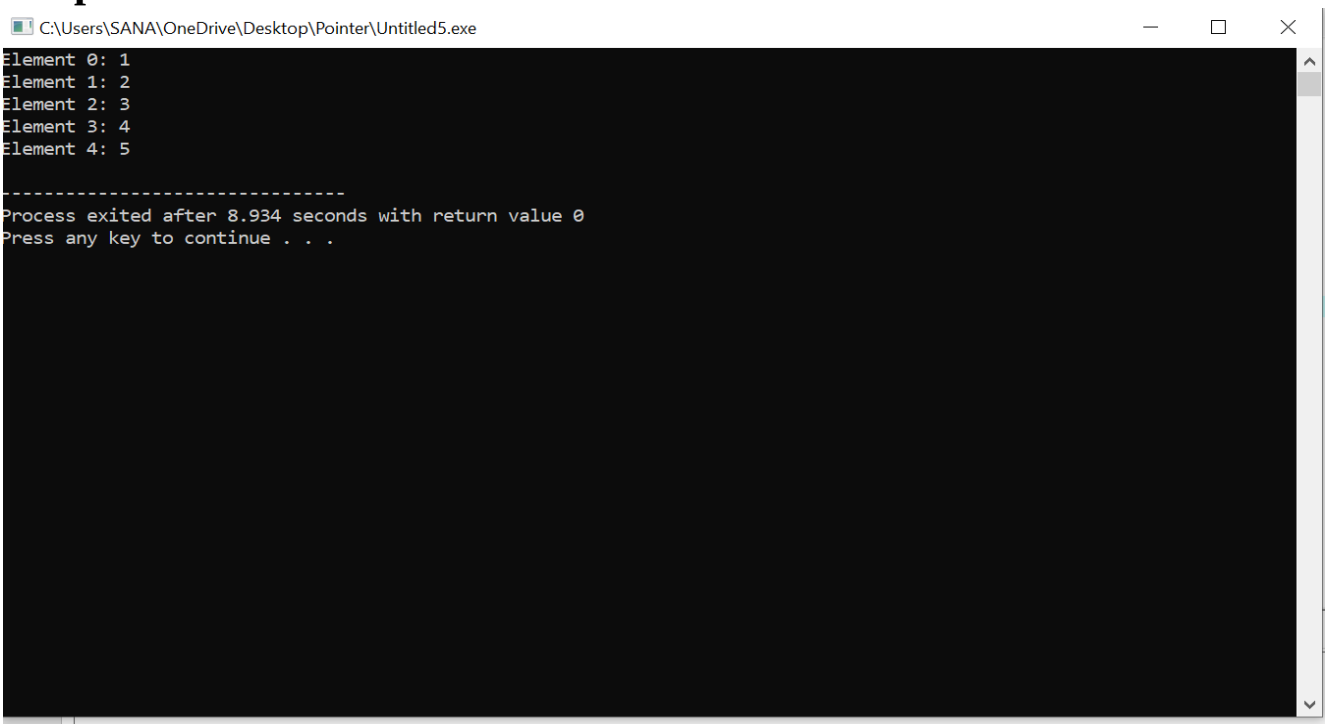


Problem:5

```
#include <iostream>


int main() {
    int numbers[] = {1, 2, 3, 4, 5};
    int* ptr = numbers;


    for (int i = 0; i < 5; i++) {
        std::cout << "Element " << i << ": " << *(ptr + i) << std::endl;
    }
```

```
    return 0;
}
```

Output:

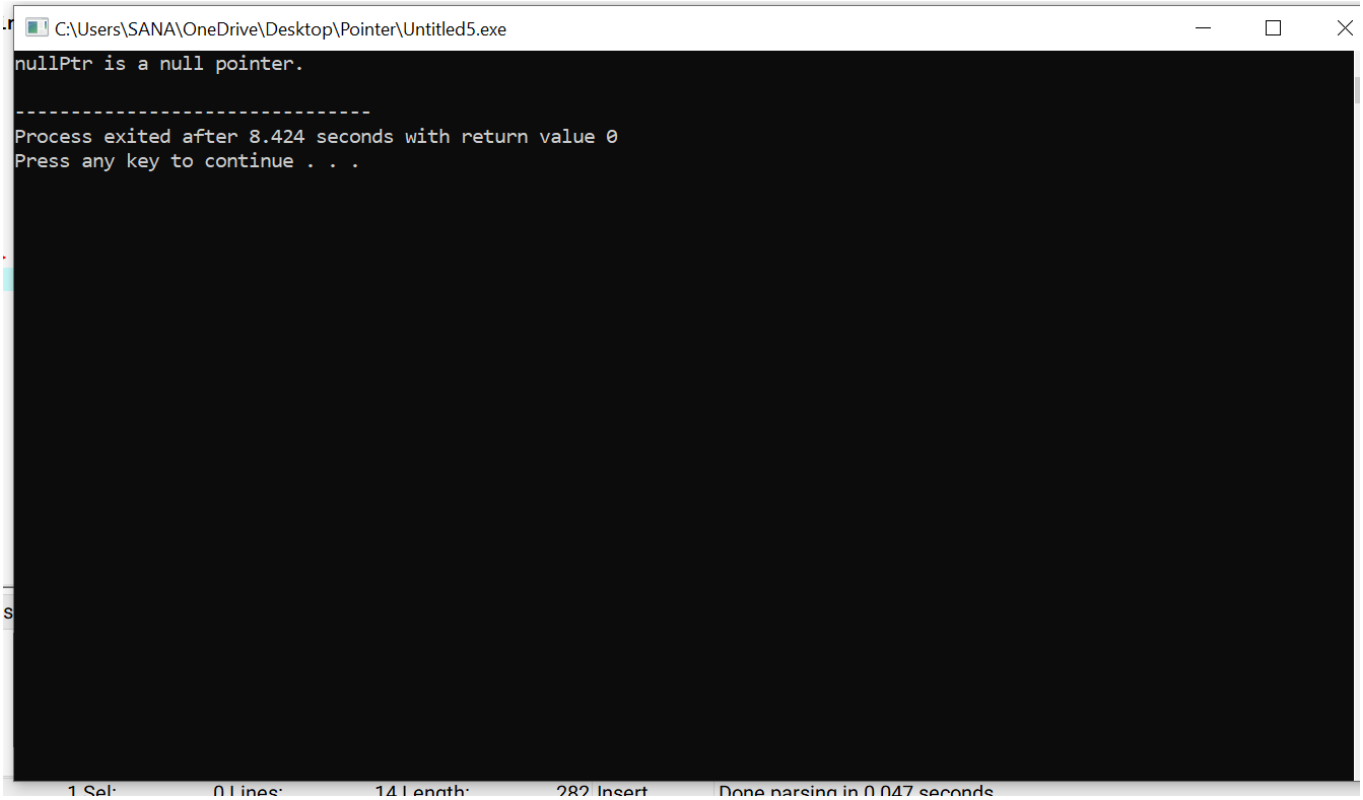


Problem: 6

```
#include <iostream>

int main() {
    int* nullPtr = nullptr;

    if (nullPtr == nullptr) {
        std::cout << "nullPtr is a null pointer." << std::endl;
    } else {
        std::cout << "nullPtr is not a null pointer." << std::endl;
    }
```

```
    return 0;
}
```

Output:

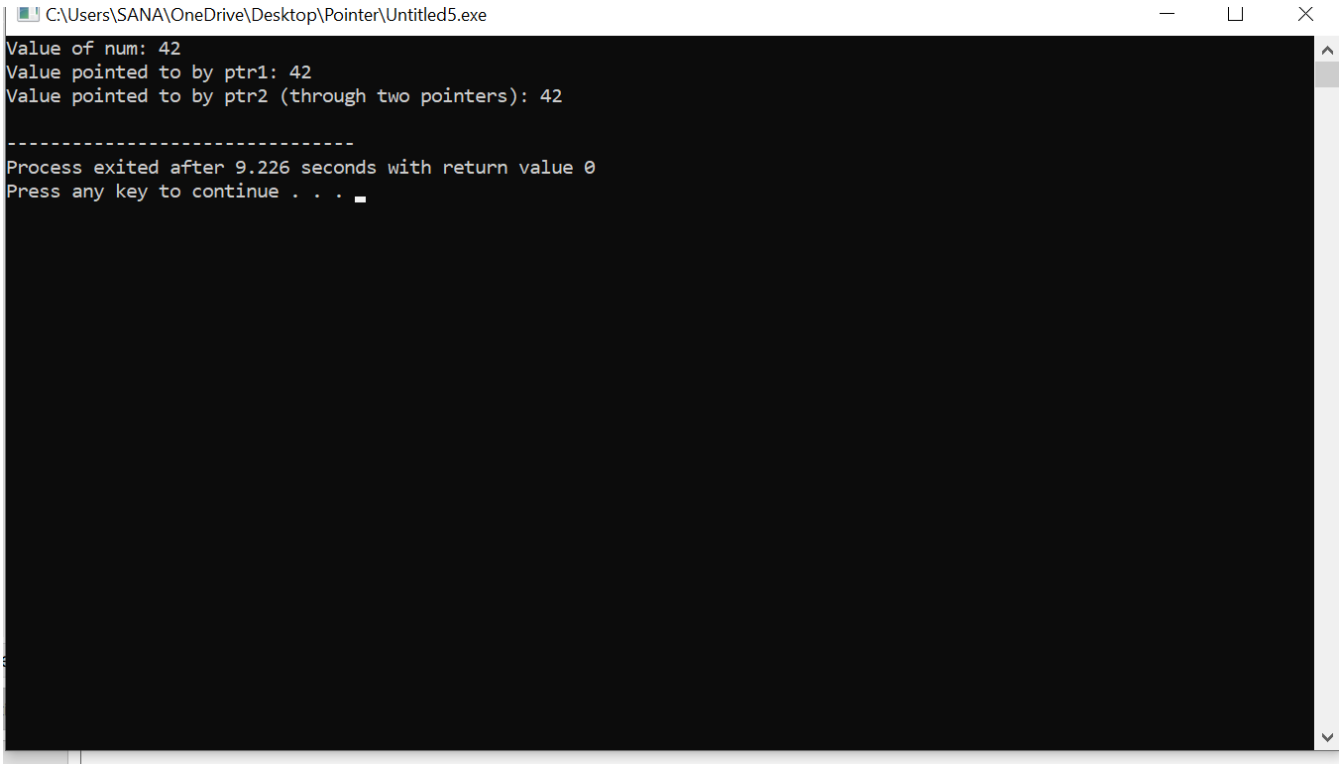


Problem:7

```
#include <iostream>

int main() {
    int num = 42;
    int* ptr1 = &num;
    int** ptr2 = &ptr1;

    std::cout << "Value of num: " << num << std::endl;
```

```
    std::cout << "Value pointed to by ptr1: " << *ptr1 << std::endl;
    std::cout << "Value pointed to by ptr2 (through two pointers): " << **ptr2 << std::endl;

    return 0;
}
```

Output:

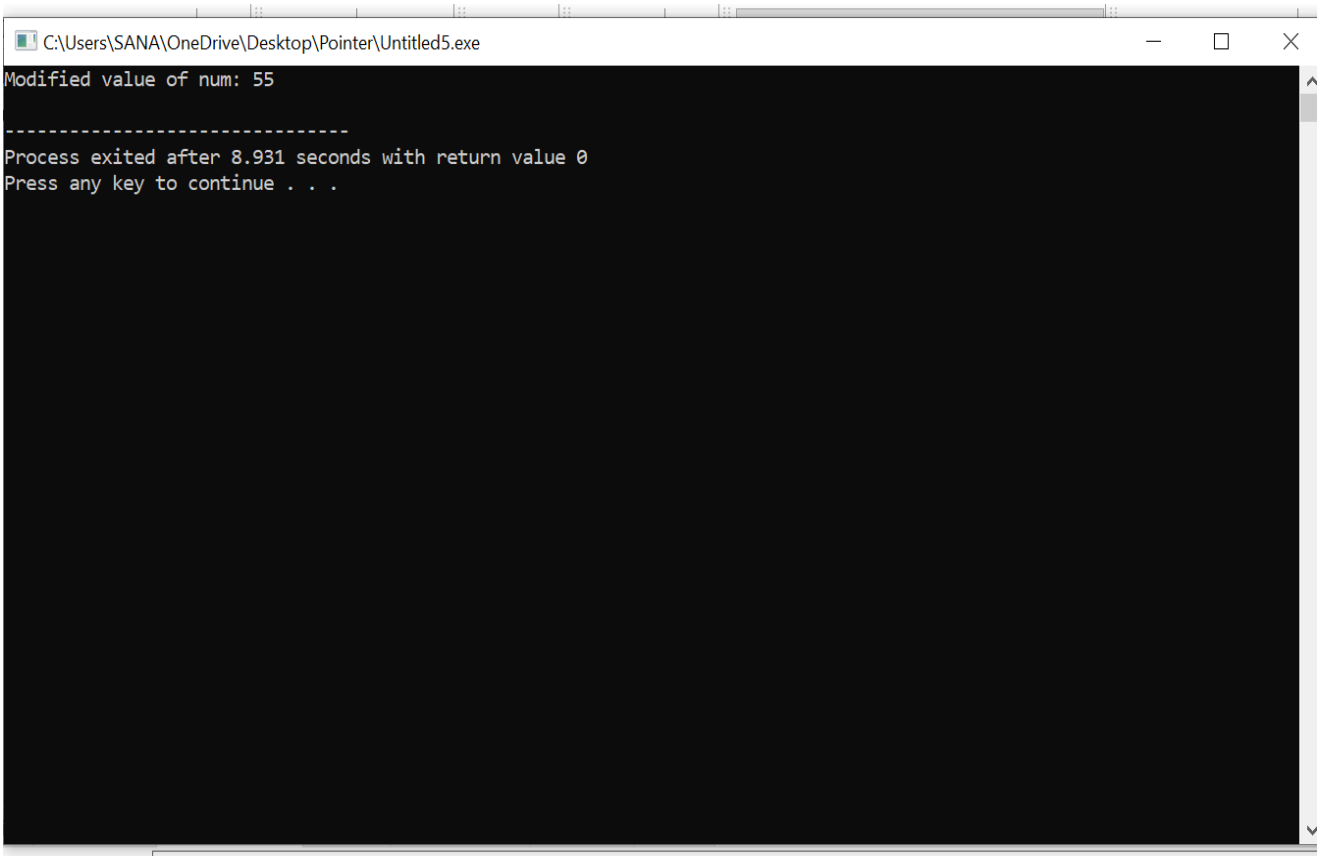


Problem: 8

```
#include <iostream>

void modifyValue(int* ptr) {
    *ptr = 55;
}
```

```
int main() {
    int num = 42;
    modifyValue(&num);
    std::cout << "Modified value of num: " << num << std::endl;
    return 0;
}
```

Output:

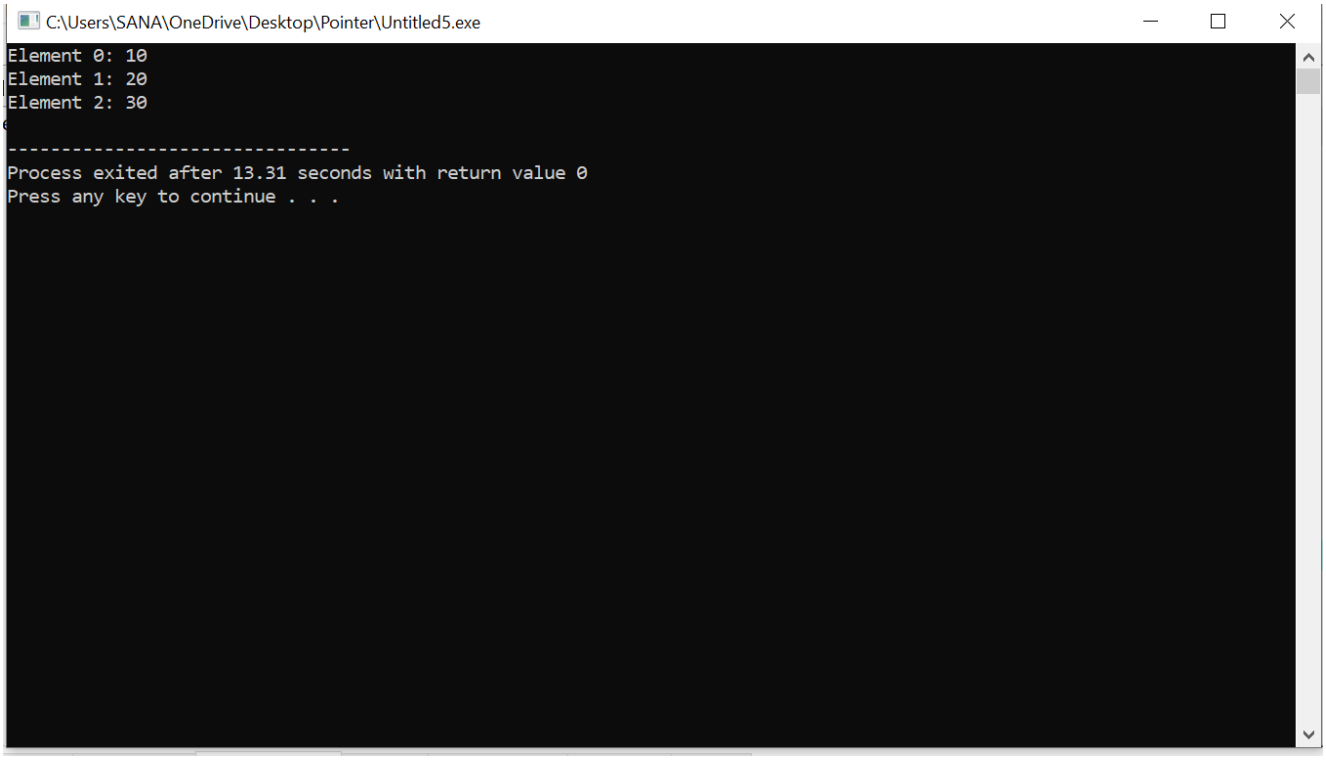


Problem: 9

```
#include <iostream>

int main() {
    int num1 = 10, num2 = 20, num3 = 30;
    int* arr[3] = {&num1, &num2, &num3};
```

```
    for (int i = 0; i < 3; i++) {
        std::cout << "Element " << i << ": " << *arr[i] << std::endl;
    }
    return 0;
}
```

Output:

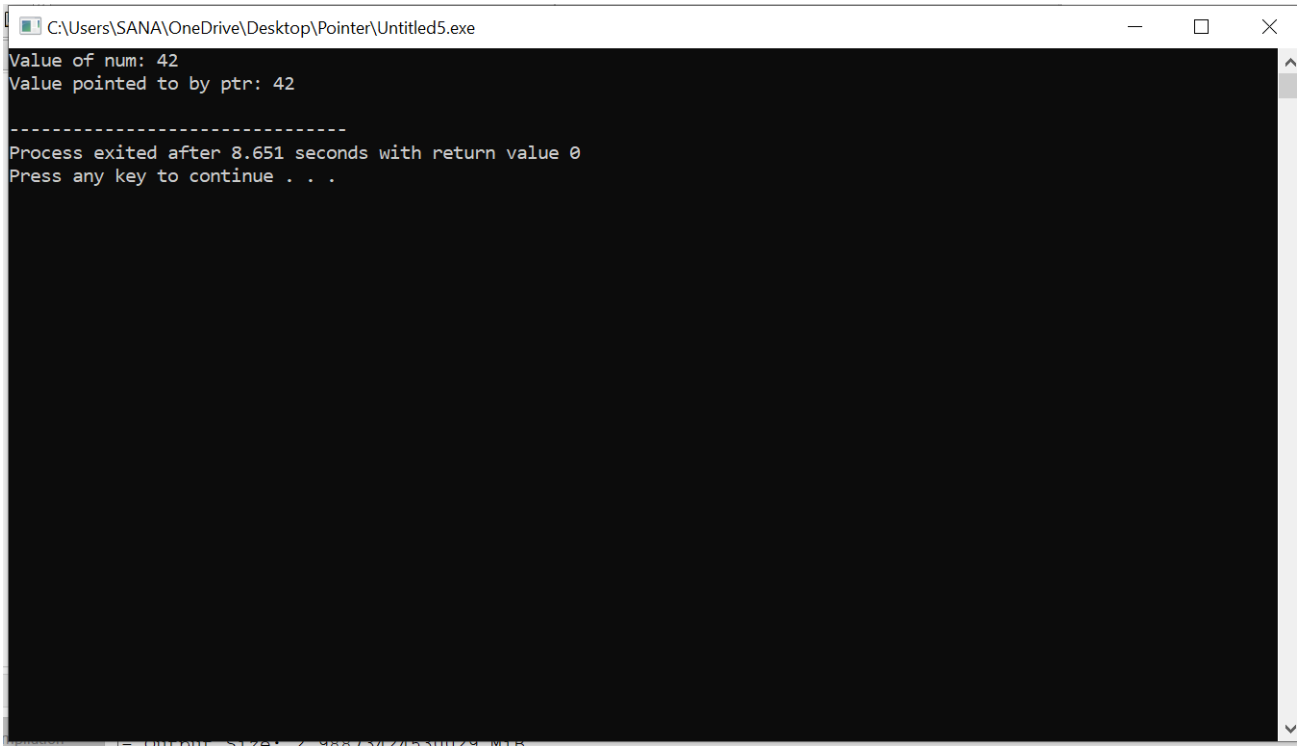


Problem: 10

```
#include <iostream>

int main() {
    const int num = 42;
    const int* ptr = &num; // Pointer to a constant integer
```

```
    std::cout << "Value of num: " << num << std::endl;
    std::cout << "Value pointed to by ptr: " << *ptr << std::endl;

    // Uncommenting the line below would result in a compilation error.
    // *ptr = 55; // Error: Cannot modify a constant through a pointer

    return 0;
}
```

Output:

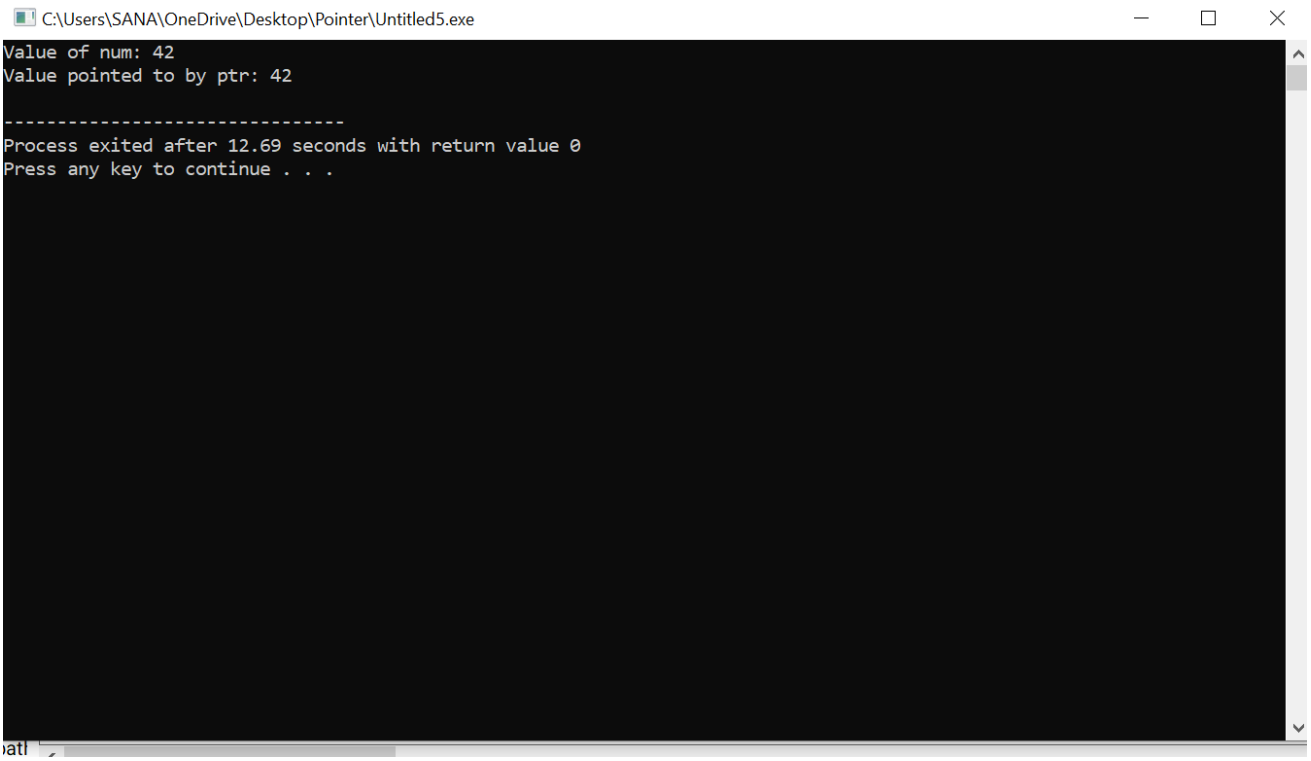


Problem:11

```
#include <iostream>

int main() {
    int num = 42;
```

```
    int* const ptr = &num; // Constant pointer to an integer

    std::cout << "Value of num: " << num << std::endl;
    std::cout << "Value pointed to by ptr: " << *ptr << std::endl;

    // Uncommenting the line below would result in a compilation error.
    // ptr = nullptr; // Error: Cannot change the value of a constant pointer

    return 0;
}
```

Output:

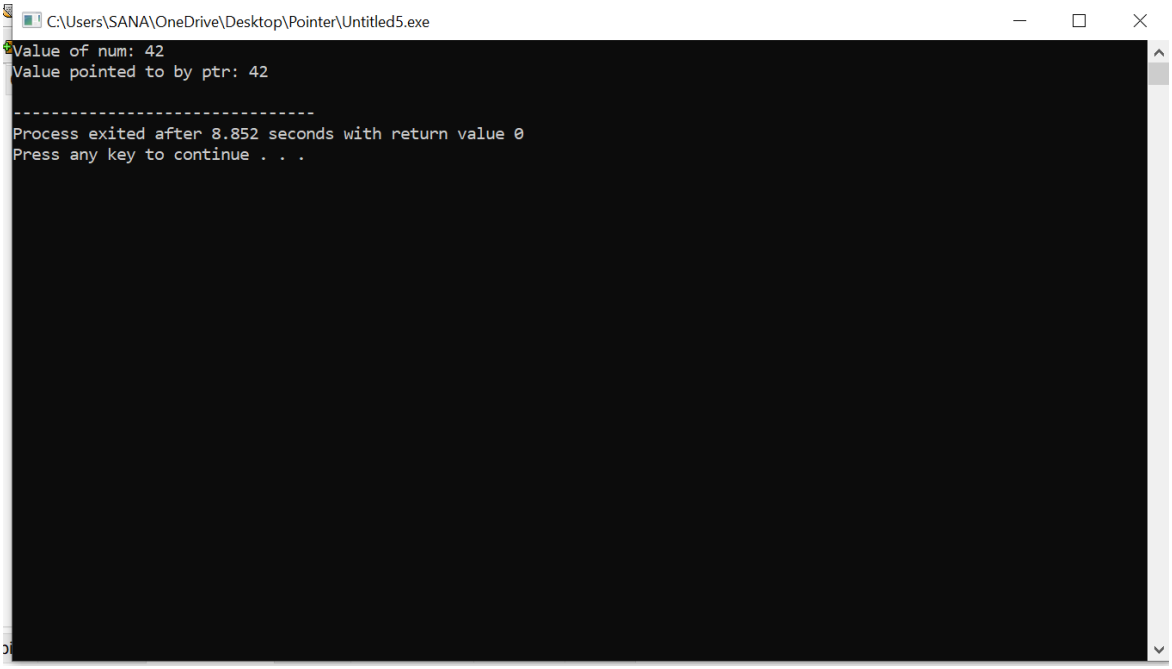


Problem: 12

```
#include <iostream>
```

```
int main() {
    const int num = 42;
    const int* const ptr = &num; // Constant pointer to a constant integer

    std::cout << "Value of num: " << num << std::endl;
    std::cout << "Value pointed to by ptr: " << *ptr << std::endl;

    // Uncommenting either of the lines below would result in a compilation error.
    // *ptr = 55; // Error: Cannot modify a constant through a pointer
    // ptr = nullptr; // Error: Cannot change the value of a constant pointer

    return 0;
}
```

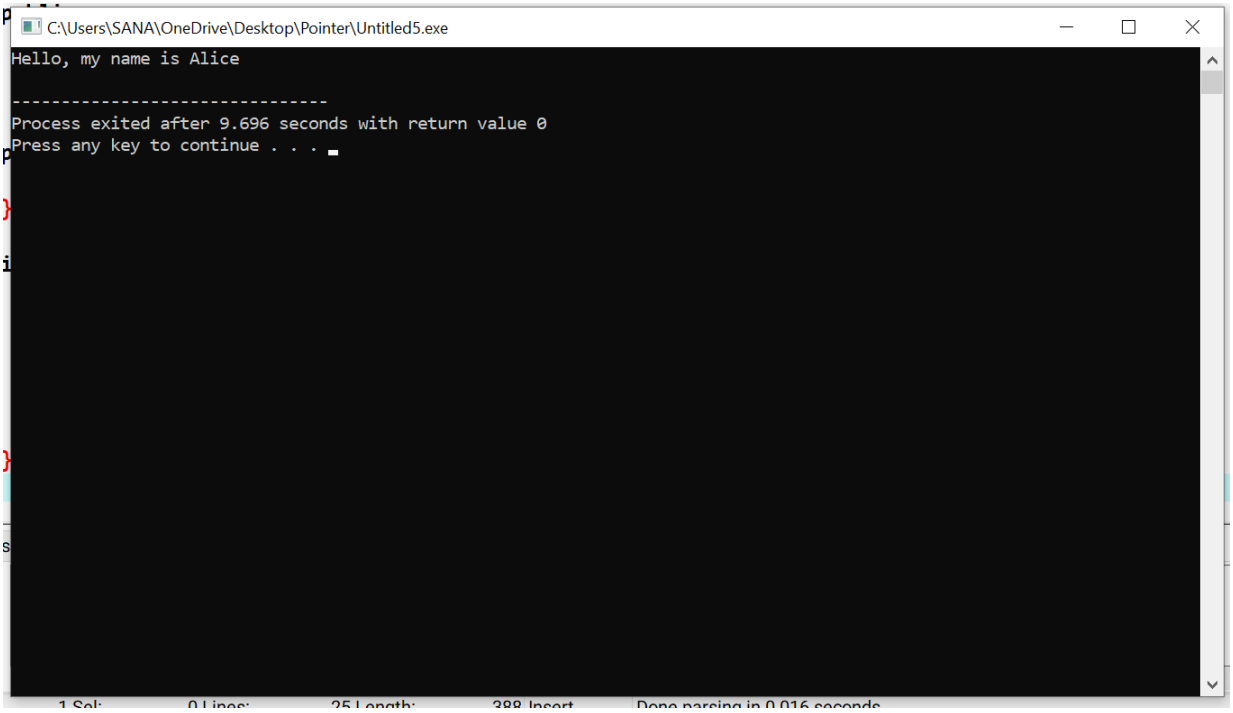
Output:



Problem:13

```
#include <iostream>
#include <string>

class Person {
public:
    Person(const std::string& name) : name_(name) {}
    void introduce() {
        std::cout << "Hello, my name is " << name_ << std::endl;
    }
private:
    std::string name_;
};

int main() {
    Person person("Alice");
    Person* ptr = &person;

    ptr->introduce();

    return 0;
}
```

Output:

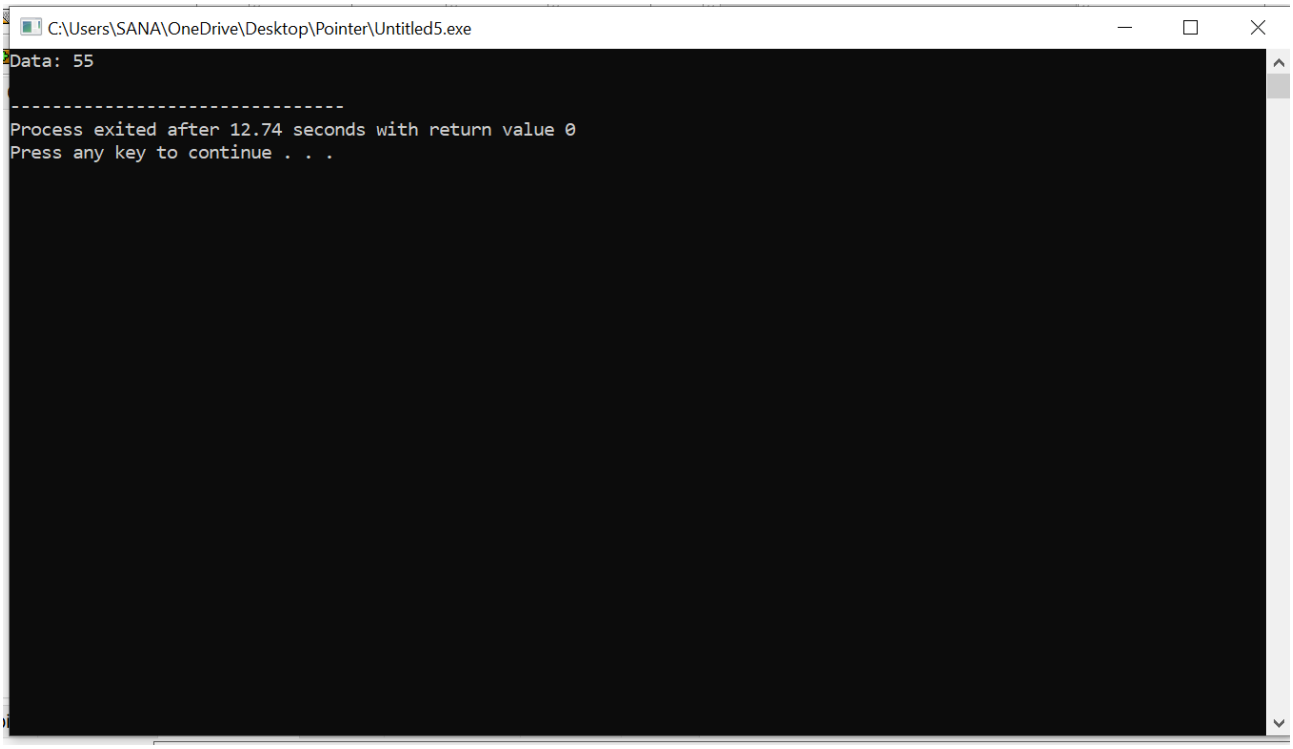


Problem: 14

```
#include <iostream>

class MyClass {
public:
    int data = 42;
    void printData() {
        std::cout << "Data: " << data << std::endl;
    }
};

int main() {
    MyClass obj;
    MyClass* ptr = &obj;
```

```
    ptr->data = 55;
    ptr->printData();

    return 0;
}
```

Output:



Problem: 15

```
#include <iostream>

class Base {
public:
    virtual void show() {
        std::cout << "This is the Base class." << std::endl;
    }
```

```
};

class Derived : public Base {
public:
    void show() override {
        std::cout << "This is the Derived class." << std::endl;
    }
};

int main() {
    Base baseObj;
    Derived derivedObj;
```

Output: