FEU INSTITUTE OF TECHNOLOGY

COLLEGE OF COMPUTER STUDIES AND MULTIMEDIA ARTS

CS0045

(COMPUTER GRAPHICS AND VISUALIZATION)

EXERCISE

2

REVIEW OF C++ PROGRAMMING

FEU INSTITUTE OF TECHNOLOGY

Student Name / Group Name:	
Members (if Group):	**Name** / **Role** DEL ROSARIO, Jerome PINZA, Judinele
Section:	TN31
Professor:	Ms. Kim Giselle A. Bautista

Objectives

1. **To reinforce understanding of C++ fundamentals including functions, arrays, and pointers.**
2. **To develop problem-solving skills by completing partially written C++ programs.**
3. **To practice writing syntactically correct C++ code in a real development environment (Visual Studio).**
4. **To encourage independent and collaborative learning by solving incremental programming challenges.**

Learning Outcomes (LO)

After completing this activity, students will be able to:

- **LO1: Identify missing code components in function, array, and pointer programs.**
- **LO2: Apply knowledge of function prototypes, parameters, return values, and nested calls in C++.**
- **LO3: Demonstrate correct usage of arrays and pointers, including pointer arithmetic and dynamic memory.**

- **LO4: Implement basic problem solutions in Visual Studio and debug simple errors.**
- **LO5: Integrate functions, arrays, and pointers to build small modular programs.**

Program Learning Outcomes (PLO)

This activity contributes to the achievement of the following PLOs (based on standard Computing/IT outcomes):

- **PLO1 – Knowledge for Solving Computing Problems: Apply knowledge of mathematics, science, and computing fundamentals to solve well-defined computing problems.**
- **PLO2 – Problem Analysis: Identify, formulate, and analyze problems, and select appropriate computing techniques to solve them.**
- **PLO3 – Design/Development of Solutions: Design and implement solutions for basic computing problems using structured programming.**
- **PLO5 – Modern Tool Usage: Apply appropriate programming tools (Visual Studio) to execute, test, and debug code.**
- **PLO9 – Individual and Team Work: Demonstrate the ability to work effectively as an individual and as a member of a team when solving programming tasks.**

C++ Programming Problems

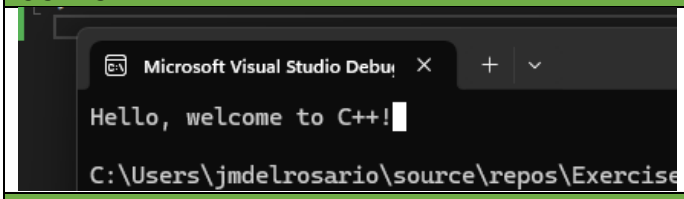
A. Part 1 – Functions

1.) Basic Function

- Write a function void ***greet()*** that simply prints "Hello, welcome to C++!".

- Call it from ***main().***

OUTPUT:



SOURCE CODE:

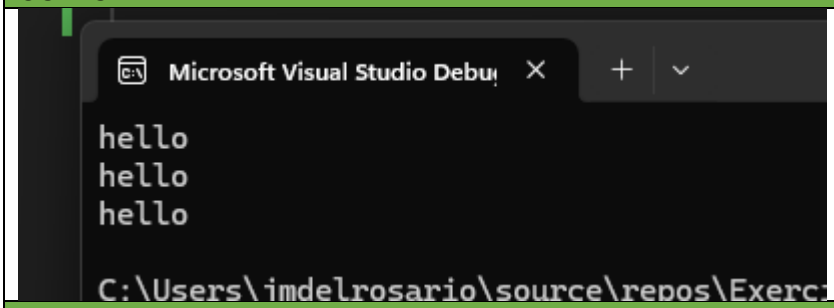
```
#include <iostream>
```

```
using namespace std;
void greet() {
    cout << "Hello, welcome to C++!" << endl;
}

int main() {
    greet();   // call the function
    return 0;
}
```

2.) Function with Parameters

- Create a function void *repeatMessage(string msg, int n)* that prints the message n times.

OUTPUT:



SOURCE CODE:

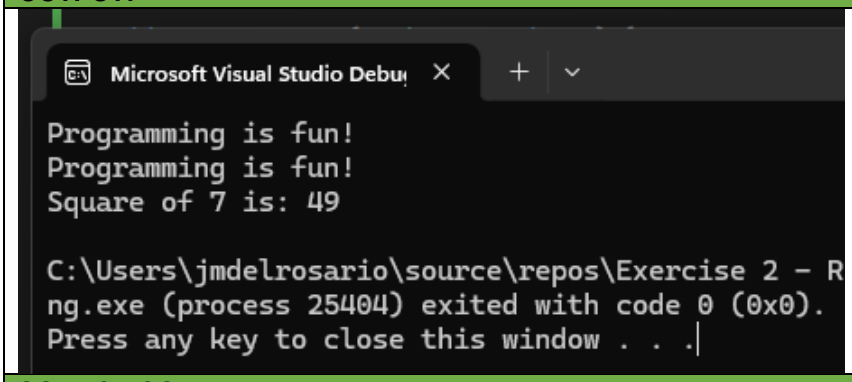
```
#include <iostream>
using namespace std;
void repeatMessage(string msg, int n) {
    for (int i = 0; i < n; i++) {
        cout << msg << endl;
    }
}
int main() {
    repeatMessage("hello", 3);
    return 0;
}
```

3.) Function with Return Value

- Implement int *square(int x)* that returns the square of x. Test it in main().

Note! Ms. Kim, I included the repeatMessage since I don't if I should keep it or remove it.

OUTPUT:



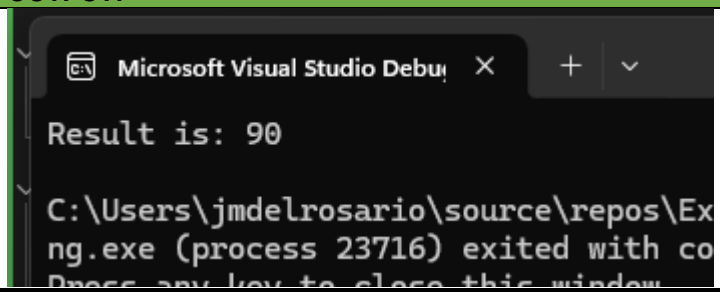
SOURCE CODE:

```
#include <iostream>
using namespace std;

void repeatMessage(string msg, int n) {
    for (int i = 0; i < n; i++) {
        cout << msg << endl;
    }
}

int square(int x) {
    return x * x;
}

int main() {
    repeatMessage("Programming is fun!", 2);
    cout << "Square of 7 is: " << square(7) << endl;
    return 0;
}
```

4.) Nested Function Calls

- Write int *add(int a, int b)* and int multiply(int a, int b).
- *Then compute (add(5, 10)) * multiply(2, 3) in main().*

OUTPUT:



SOURCE CODE:

```
#include <iostream>
using namespace std;

int add(int a, int b) {
    return a + b;
}

int multiply(int a, int b) {
    return a * b;
}

int main() {
    int result = add(5, 10) * multiply(2, 3);
    cout << "Result is: " << result << endl;
    return 0;
}
```

1. Basic Functions

```
#include <iostream>

using namespace std;

___ greet() {

    cout << "Hello, welcome to C++!" << endl;

}


int main() {

    ___();   // call the function

    return 0;

}
```

2. Function with Parameters

```
#include <iostream>
```

```
using namespace std;

void repeatMessage(___ msg, ___ n) {
    for (int i = 0; i < n; i++) {
        cout << msg << endl;
    }
}
```

3. Function with Return Value

```
int main() {
    repeatMessage("Programming is fun!", ___);
    return 0;
}
#include <iostream>
using namespace std;

int square(___ x) {
    return ___ * ___;
}

int main() {
    cout << "Square of 7 is: " << square(___) << endl;
    return 0;
}
```

4. Nested Function Calls

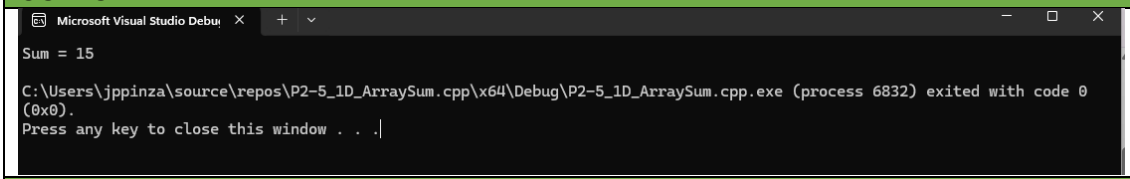
```
#include <iostream>

using namespace std;


int add(int a, int b) {

    return ___;

}


int multiply(int a, int b) {

    return ___;

}


int main() {

    int result = add(5, 10) * multiply(___, ___);

    cout << "Result is: " << _______<< endl;

    return 0;

}
```

B. Part 2 – Arrays

5.) 1D Array Sum

- Create an array of 5 integers.
- Write a function int *sumArray(int arr[], int size)* that returns the sum.

OUTPUT:



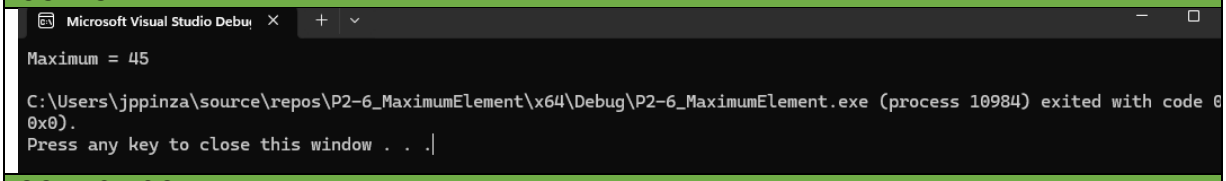
SOURCE CODE:

```
#include <iostream>
using namespace std;

int sumArray(int arr[], int size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += arr[i];
    }
    return sum;
}

int main() {
    int numbers[5] = { 1, 2, 3, 4, 5 };
    cout << "Sum = " << sumArray(numbers, 5) << endl;
    return 0;
}
```

6.) Maximum Element

- Write a function int *findMax(int arr[], int size)* that finds the maximum element in an array.

OUTPUT:



SOURCE CODE:

```
#include <iostream>
using namespace std;

int findMax(int arr[], int size) {
    int maxVal = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] > maxVal) {
            maxVal = arr[i];
        }
    }
    return maxVal;
}

int main() {
    int nums[5] = { 12, 45, 7, 23, 34 };
    cout << "Maximum = " << findMax(nums, 5) << endl;
    return 0;
}
```

```
}
```

5.) 1D Array Sum

```
#include <iostream>
using namespace std;

int sumArray(int arr[], int size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum ___ arr[i];
    }
    return sum;
}

int main() {
    int numbers[5] = {1, 2, 3, 4, 5};
    cout << "Sum = " << ___(numbers, 5) << endl;
    return 0;
}
```

6.) Maximum Element

```
#include <iostream>
using namespace std;

int findMax(int arr[], int size) {
    int maxVal = arr[0];
```

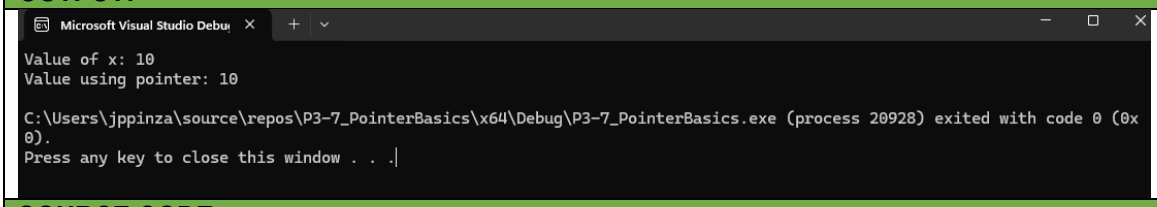
```
    for (int i = 1; i < size; i++) {
        if (arr[i] ___ maxVal) {
            maxVal = ___;
        }
    }
    return maxVal;
}

int main() {
    int nums[5] = {12, 45, 7, 23, 34};
    cout << "Maximum = " << findMax(nums, ___) << endl;
    return 0;
}
```

C. Part 3 – Pointers

7. Pointer Basics

- Declare an ***int x = 10; int* p = &x;.***
- Print the value of x using both ***x*** and ****p.***

OUTPUT:



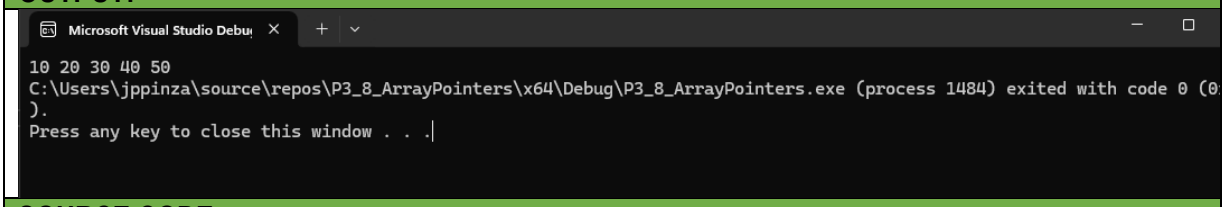
SOURCE CODE:

```
#include <iostream>
using namespace std;

int main() {
    int x = 10;
    int* p = &x;

    cout << "Value of x: " << x << endl;
    cout << "Value using pointer: " << *p << endl;
    return 0;
}
```

8. Array and Pointers

- Using pointer arithmetic, print the elements of an array (without using []).

OUTPUT:



SOURCE CODE:

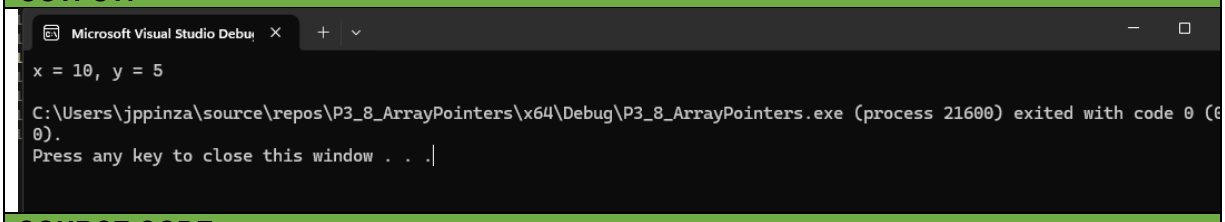
```
#include <iostream>
using namespace std;

int main() {
    int arr[5] = { 10, 20, 30, 40, 50 };
    int* p = arr;   // array name acts like pointer

    for (int i = 0; i < 5; i++) {
        cout << *(p + i) << " ";
    }
```

```
    return 0;
}
```

9. Swap Function (by pointer)

- Write a function void *swap(int* a, int* b)* that swaps two integers using pointers.

OUTPUT:

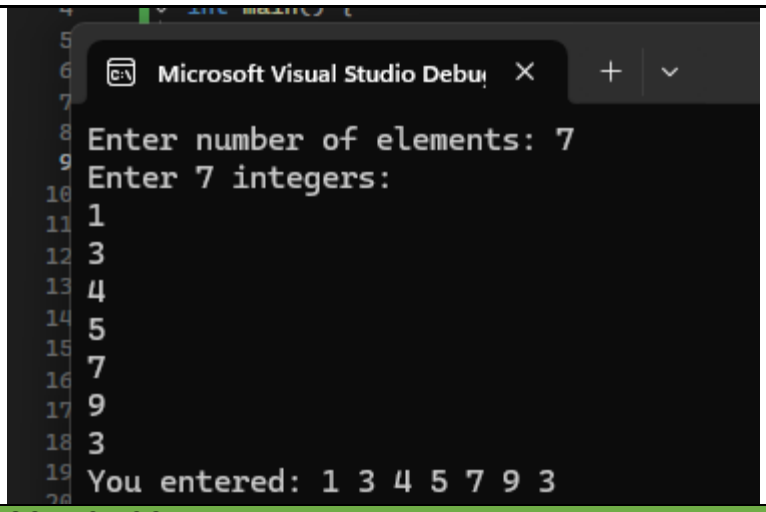


SOURCE CODE:

```
#include <iostream>
using namespace std;

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int main() {
    int x = 5, y = 10;
    swap(&x, &y);
    cout << "x = " << x << ", y = " << y << endl;
    return 0;
}
```

10. **Dynamic Memory**
    a. Write a program that:
        i. Asks the user for n.
        ii. Dynamically allocates an array of size n.
        iii. Stores values, prints them, and deletes the array.

OUTPUT:



SOURCE CODE:

```
#include <iostream>
using namespace std;

int main() {
    int n;
    cout << "Enter number of elements: ";
    cin >> n;

    int* arr = new int[n];

    cout << "Enter " << n << " integers:\n";
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }
    cout << "You entered: ";
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;

    delete[] arr;
    return 0;
}
```

7.) Pointer Basics

```
#include <iostream>

using namespace std;
```

```
int main() {
    int x = 10;
    int* p = ___;

    cout << "Value of x: " << x << endl;
    cout << "Value using pointer: " << ___ << endl;
    return 0;
}
```

8.) Array and Pointers

```
#include <iostream>
using namespace std;

int main() {
    int arr[5] = {10, 20, 30, 40, 50};
    int* p = ___;   // array name acts like pointer

    for (int i = 0; i < 5; i++) {
        cout << ___ << " ";
    }
    return 0;
}
```

9.) Swap Function

```
#include <iostream>
using namespace std;

void swap(___ a, ___ b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int main() {
    int x = 5, y = 10;
    swap(&x, &y);
    cout << "x = " << x << ", y = " << y << endl;
    return 0;
}
```

10.) Dynamic Memory

```
#include <iostream>
using namespace std;

int main() {
    int n;
    cout << "Enter number of elements: ";
    cin >> n;

    int* arr = ___ int[n];  // allocate memory
```

```
    for (int i = 0; i < n; i++) {
        cin >> ___[i];
    }

    cout << "You entered: ";
    for (int i = 0; i < n; i++) {
        cout << ___[i] << " ";
    }

    ___ arr;  // free memory
    return 0;
}
```