Final Exam - Fall 2024

National University of Computer and Emerging Sciences

Compiled by Arqum Mehtab CS-D 25

Table of contents

1 Part 1: Multiple Choice Questions

1. Which of the following is true about the `#include` directive in C++?

- A. It includes standard libraries only.
- B. It is used to include both standard and user defined header files.
- C. It can only include headers from the standard library.
- D. It must be placed at the beginning of a program.

2. Which of the following demonstrates an implicit type conversion in C++?

- A. `int x; double y = x;`
- B. `int x = (int) 3.5;`
- C. `double y; static_cast<double>(5);`
- D. `float z = atof("3.14");`

3. Which of the following is a valid use of the switch statement in C++?

- A. `switch(3.14) { case 3.14: cout << "Floating Point"; }`
- B. `switch(true) { case true: cout << "Boolean"; case false: cout << "False"; }`
- C. `switch(x) { case 'a': cout << "Char"; case 97: cout << "ASCII"; }`
- D. `switch(s) { case "Hello": cout << "Greeting"; break; case "H": cout << "Character"; }`

4. Which of the following best describes the purpose of a function prototype in C++?

- A. To define the body of the function.
- B. To allow the compiler to know about the function before its actual definition.
- C. To allocate memory for the function.
- D. To call the function in the main program.

5. Which of the following statements about arrays is true?

- A. Arrays can change their size dynamically after declaration.
- B. The size of an array must be known at compile time.
- C. Arrays can hold elements of different data types.
- D. The name of an array is a pointer to its first element that cannot be changed.

6. Which of these statements about pointers is false?

- A. A pointer can store the address of a variable.
- B. Pointers can be used for dynamic memory allocation.
- C. A pointer can only point to variables of the same type.
- D. Pointers can be dereferenced to access the value at the memory location they hold.

7. You need to open a file named `data.txt` in your C++ program for both reading and writing. You want to ensure that the file is opened in such a way that you can perform both input and output operations. Which of the following code snippets correctly demonstrates how to open the file for reading and writing?

- A. `ifstream file("data.txt", ios::in);`
- B. `ofstream file("data.txt", ios::in | ios::out);`
- C. `fstream file("data.txt", ios::in | ios::out);`
- D. `istream file("data.txt", ios::in | ios::out);`

8. Which statement is correct when using pointers with arrays?

- A. An array name can be used as a pointer.
- B. You must use pointer arithmetic to access array elements.
- C. Pointers and arrays are incompatible in C++.
- D. Array elements must be individually assigned to pointers.

9. Consider the following C++ code snippet:

```
#include <iostream>
using namespace std;
int main() {
    int arr[] = {10, 20, 30, 40, 50};
    int *ptr = arr;
    cout << "First element: " << *ptr << endl; // Line 1
    cout << "Second element: " << *(ptr + 1) << endl; // Line 2
    cout << "Third element: " << ptr[2] << endl; // Line 3
    return 0;
}
```

Which of the following statements about the code above is true?

- A. `*ptr` accesses the first element of the array, `ptr[2]` accesses the third element, and `*(ptr + 1)` accesses the second element.
- B. `*ptr` accesses the second element of the array, `ptr[2]` accesses the third element, and `*(ptr + 1)` accesses the first element.
- C. The code will cause a compilation error due to the use of pointer arithmetic.
- D. The code will not compile because arrays and pointers are incompatible in C++.

10. What happens when memory dynamically allocated using `new` is not deallocated using `delete`?

- A. Compilation error.
- B. The allocated memory is automatically freed by the OS.
- C. Memory leaks occur as the memory is not properly released.
- D. The program will not run.

11. Consider the following C++ code snippet:

```
#include <iostream>
using namespace std;
int main() {
    for (int i = 0; i < 100000; i++) {
        int *ptr = new int[1000];
        ptr[0] = i;
    }
    return 0;
}
```

What is the issue with the code above?

- A. The code has a compilation error due to incorrect memory allocation syntax.
- B. The dynamically allocated memory is automatically freed when the loop ends.
- C. Memory leaks occur as the dynamically allocated memory is not deallocated within the loop.
- D. The program will run normally without any issue, but it will consume more memory over time.

12. Which of the following correctly passes an array to a function in C++?

- A. `void func(int arr[]) { ... }`
- B. `void func(int* arr) { ... }`
- C. Both A and B.
- D. Neither A nor B.

13. Consider the following C++ code:

```
#include <iostream>
using namespace std;
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}
int main() {
    int nums[] = {1, 2, 3, 4, 5};
    printArray(nums, 5);
    printArray(&nums[0], 5);
    return 0;
}
```

What will happen in this code?

- A. The first call to `printArray(nums, 5)` will correctly print the array elements, but the second call will cause an error because arrays cannot be passed by pointer.
- B. Both calls to `printArray(nums, 5)` and `printArray(&nums[0], 5)` will correctly print the array elements as they are both valid ways to pass the array.
- C. The second call to `printArray(&nums[0], 5)` will cause a segmentation fault because the pointer to the first element is being passed incorrectly.
- D. The first call to `printArray(nums, 5)` will print the array elements, but the second call will print garbage values due to incorrect pointer usage.

14. Which of the following statements is true for reading from a file in C++?

- A. The `eof()` function returns true if the end of the file is reached.
- B. The `read()` function is used to read a single character.
- C. The `ignore()` function is used to skip lines.

- D. The `getline()` function can only read lines up to 1024 characters.

15. What will be the output of the following code if `f(1)` is called?

```
int f(int x) {
    if (x == 0) return 0;
    else return x + f(x - 1);
}
```

- A. 0
- B. 1
- C. 2
- D. -1

16. What will be the output of the following code?

```
int a = 5, b = 10, c = 2;
if (a > b && a > c)
// [Code snippet cuts off here in image]
```

17. What will be the output of the following code snippet:

```
bool flag = true;
flag = !flag;
cout << flag << endl;
```

- A. 1
- B. 0
- C. true
- D. false

18. Consider the following C++ code:

```
#include <iostream>
using namespace std;
void greet(string name, string greeting = "Hello", int times = 1) {
    for (int i = 0; i < times; i++) {
        cout << greeting << ", " << name << "!" << endl;
    }
}
int main() {
    greet("Alice");
    greet("Bob", "Hi");
    greet("Charlie", "Good morning", 3);
    return 0;
}
```

What will happen in the code?

- A. The function greet can only be called with all arguments specified, including the default arguments, because all arguments need to have default values.
- B. Default arguments must be provided in the function definition, so the code will work correctly.
- C. Default arguments must always be specified as the trailing arguments in the function, meaning they should come after non-default arguments.
- D. Default arguments can be specified anywhere in the function parameter list, and the code will still work correctly.

19. Which of the following operators has the highest precedence in C++?

- A. `++` (increment)
- B. `*` (multiplication)

- C. `&&` (logical AND)
- D. `==` (equality)

20. Which of the following correctly demonstrates a combined assignment operator?

- A. `int result = a * b + c;`
- B. `result = result * 2;`
- C. `result *= 2;`
- D. `result << 2;`

21. What is the correct order of associativity for the following operators: `+=`, `-`, `*`, `++`?

- A. `*` (left), `-` (left), `*` (left), `++` (right)
- B. `+=` (right), `*` (left), `-` (left), `++` (left)
- C. `+=` (right), `-` (left), `*` (right), `++` (left)
- D. `*` (left), `-` (right), `+=` (left), `++` (right)

22. Which of the following expressions evaluates to true if `x = 10`, `y = 20`, and `z = 30`?

- A. `(x > y) && (y < z)`
- B. `(x > y) || (z < x)`
- C. `!(x == 10)`
- D. `(x >= 10) && (y <= 10) && (z == 30)`

23. Which of the following logically combines two conditions A and B such that the result is true only if both A and B are false?

- A. `!(A || B)`
- B. `A && B`
- C. `!A || !B`
- D. `A || B`

24. Which statements correctly access the third element of an integer array `arr` using a double pointer?

- A. `*(*(arr) + 2)`
- B. `*(*(arr + 2) + 1)`
- C. `*(*(arr + 3) - 1)`
- D. `*(*(arr) + 3)`

25. In the expression: `a + (b * c - d) / e % f`, which operation is performed first?

- A. Addition (+)
- B. Multiplication (*)
- C. Modulus (%)
- D. Subtraction (-)

26. Which of the following is false about function overloading in C++?

- A. Functions can be overloaded by having different return types.
- B. Functions can be overloaded by having different parameter types.
- C. Functions can be overloaded by having a different number of parameters.
- D. Functions can be overloaded within the same scope.

27. Which of the following best describes recursion in C++?

- A. A function calling another function.
- B. A function calling itself repeatedly.
- C. A loop inside a function.
- D. A function with multiple return statements.

28. How does C++ handle variables with the same name in different scopes?

- A. The variable in the innermost scope hides the outer scope variable.
- B. An error is generated at compile time.
- C. Both variables are accessible simultaneously.
- D. The variable in the outer scope hides the inner scope variable.

29. Which of the following correctly demonstrates the use of a constant variable in C++?

- A. `const int a = 5;`
- B. `int const a = 5;`
- C. `a = 5;`
- D. Both a and b

30. Can a class in C++ inherit privately from one class and publicly from another?

- A. Yes, with no restrictions.
- B. No, all inheritance must have the same visibility.
- C. Yes, but only if both classes are virtual base classes.
- D. Yes, but it affects member accessibility.

31. How does C++ handle different data types in a single expression?

- A. By promoting all data types to the highest precision type.
- B. By demoting all data types to the lowest precision type.
- C. By using implicit type conversion.
- D. By using explicit type conversion.

32. What is the correct way to read a line of text from a file into a string variable in C++?

- A. `file >> line;`
- B. `getline(file, line);`
- C. `file.getline(line);`
- D. `getline(line, file);`

33. Which of the following functions checks if a file stream has reached the end of the file?

- A. `eof()`
- B. `eof`
- C. `is_end`
- D. `end_of`

34. Which of the following is NOT a valid justification for using function overloading in C++?

- A. To allow functions with the same name to handle different types of arguments.
- B. To extend the functionality of operators.
- C. To replace macro definitions.
- D. To improve code readability and maintenance by allowing similar functions to share the same name.

35. When using the `new` operator to allocate memory dynamically for a data type in C++, the `delete` operator must be used to deallocate the memory. Failure to do so will lead to...

- A. Syntax errors
- B. Memory leaks
- C. Logic errors
- D. Compilation errors

36. Which of the following expressions would result in true if `x = 10` and `y = 20`?

- A. `!x && y`
- B. `x > y || x <= y`
- C. `x == 10 && !y`
- D. `x == 10 || !(y < 15)`

37. Which of the following statements about dynamic memory allocation is false?

- A. The new operator allocates memory on the heap.
- B. The delete operator deallocates memory on the stack.
- C. Memory leaks occur if dynamically allocated memory isn't properly deallocated.
- D. The delete operator is used for both single objects and arrays.

38. Which of the following will correctly check for dynamic memory allocation failure?

- A. `if (!ptr) cout << "Allocation failed";`
- B. `if (ptr < 0) cout << "Allocation failed";`
- C. `if (ptr == NULL) cout << "Allocation failed";`
- D. `if (ptr == nullptr) cout << "Allocation failed";`

39. What does the following code do?

```
int *p;
int x = 10;
p = &x;
```

- A. Declares a pointer p and initializes it with the address of x.
- B. Declares a pointer p and assigns 10 to x.
- C. Declares an integer variable p and assigns x to 10.
- D. Declares a pointer p and assigns x to p.

40. What is the logical error in the following code snippet?

```
int findMax(int a, int b, int c) {
    if (a > b && a > c)
        return a;
    else if (b > a && b > c)
        return b;
    else
        return c;
}
```

- A. The function will not compile because of a syntax error.
- B. The function findMax will always return c if two values are equal.
- C. The function does not handle cases where all three numbers are equal.
- D. The code produces a runtime error if all three values are the same.

2 Part 2: Coding Questions

2.1 Question 2 [24 Marks]

Write the output of the following C++ codes (if the code is correct). If you find any error/s in the code, please identify and explain the error/s.

Code 1

```
const char* x[] = {"final", "Programming", "Fundamentals", "exam"};
char const ** xy[] = {x+3, x+1, x+2, x};
char const *** xyz[] = {xy+2, xy+3};
int main() {
    cout << ***xyz[0] << endl;
    cout << (**xyz)[-1][0] << endl;
    cout << (*xy)[-1] << endl;
    cout << (*(xyz[1][-1]) + 3) << endl;
```

```
    return 0;
}
```

Code 2

```
int a = 25;
int* fun() {
    static int a = 25;
    ::a++;
    if(a) {
        static int a = 25;
        a--;
        cout << a << endl;
    }
    cout << a << endl;
    return &a;
}
int main() {
    fun();
    int *qp = fun();
    cout << (*qp + 1) << endl;
    cout << a;
    return 0;
}
```

Code 3

```
#include <iostream>
using namespace std;
void foo(float** ptr) {
    *ptr = new float;
    **ptr = 3.14f;
    cout << "Value from foo: " << **ptr << endl;
}
void foo2(float* ptr) {
    if (ptr) {
        (*ptr)++;
    }
    cout << "Value from foo2: " << (*ptr)++ << endl;
}
int main() {
    float* myFloat = nullptr;
    foo(&myFloat);
    if (myFloat) {
        cout << "Value from main: " << *myFloat << endl;
    }
    foo2(myFloat);
    cout << "Value from main: " << *myFloat << endl;
    delete myFloat;
    return 0;
}
```

Code 4

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

```
void makeMess(char str[]) {
    int step = 0;
    while(str[step] != '\0') {
        if(str[step] == 'S')
            // (strlen(str) calculates the number of characters... excluding null)
            str[step] += 8;
        step++;
    }
}
int main() {
    char stuff[] = "Structure-C-String-String-Strlen";
    cout << stuff << endl;
    makeMess(stuff);
    cout << stuff << endl;
    return 0;
}
```

Code 5

```
#include <iostream>
using namespace std;
int& hello(int a, int &b, int &c) {
    int x;
    b *= 10;
    x = a * b;
    c++;
    x -= c;
    return x;
}
int main() {
    int a=10, b=11, c=12, result;
    result = hello(a,b,c);
    cout << result;
    return 0;
}
```

Code 6

```
int* doSomething(int *p, int *q) {
    int *t = new int();
    *t = *p;
    *p = *q;
    *q = *t;
    return t;
}
int main() {
    int p = -10, q = 25;
    int *t = NULL;
    t = doSomething(&p, &q);
    cout << "p=" << p << endl;
    cout << "q=" << q << endl;
    cout << "t=" << *t << endl;
    return 0;
}
```

2.2 Question 3 [8 Marks]

Write a C++ function named `concatenate` that takes two `const char*` strings as input and returns a dynamically allocated `char*` string containing the concatenation of the two inputs. The function should allocate sufficient space to store the concatenated result (assume a total maximum length of 20 characters, including the null terminator). You are required to fill the provided space with the appropriate code statements. Declaration of new variables is not allowed.

Hint: You can use built-in functions from cstring library excluding '+' operator.

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

char* concatenate(const char* str1, const char* str2) {
    // Dynamically create array to hold the concatenated result
    char* result = ___________________________;

    // Give your logic here
    ----------------------------------------
    ----------------------------------------

    return result;
}

int main() {
    const char* str1 = "Hello, ";
    const char* str2 = "World!";
    char* res = concatenate(str1, str2);
    cout << "Concatenated String: " << res << endl;
    delete[] res;
    return 0;
}
```

2.3 Question 4 [8 Marks]

Consider the following code (assume that `b` and `n` are two integers) and answer the questions below:

```
int main() {
    int x = b;
    int k = n;
    int z = 1;
    while (k != 0) {
        if (k % 2 != 0)
            z = z * x;
        x = x * x;
        k = k / 2;
    }
    return 0;
}
```

1. The while loop in the above code would run **only** for positive integers i.e. if `n` is a positive integer. (True/False) **[1]**
2. After the first iteration of the while loop in the above code, the value of `k` will always be even (in the subsequent iterations, if any). (True/False) **[2]**
3. When the while loop in the above code terminates, which of the following must be true? **[2]**
   - i. $x = b^n$

- ii. $z = b^n$
- iii. $b = x^n$
- iv. $b = z^n$

4. What will be the value of `z`, `x`, and `k` at the end of the loop when `b = 10` and `n = 5`? **[3]**

2.4 Question 5 [5 Marks]

Given the following code answer the following:

```
int _2dArray[][3] = {{1}, {2, 3, 4}, {2}};
```

1. What will be the contents of `_2dArray`?
2. Why `_2dArray[][]` is not allowed for all dimensions in a multidimensional array?
3. `int _2dArray[][3] = {1, 2, 3, 4, 2};` What will be the contents of this array now?
4. If this array is stored in a 1D array (`_1dArray`), what will be stored in `_1dArray[4]`?
5. Write a function prototype which takes `_2dArray` as input but ensures that its content are not changed?

2.5 Question 6 [20 Marks]

Suppose you are creating a symmetric pattern on a 2D grid using nested loops and mathematical logic. The pattern consists of four distinct line segments that form a shape similar to a rotated W. Each line segment must be implemented in a separate function, and each function must output only its part of the pattern. The goal is to combine the outputs of all four functions to reproduce the complete pattern. The grid size is 50x50.

The code is partially written. Determine Mathematical Equations and implement the logic for the following functions:

- `drawSlope1`: Outputs the first line segment.
- `drawSlope2`: Outputs the second line segment.
- `drawSlope3`: Outputs the third line segment.
- `drawSlope4`: Outputs the fourth line segment.

Constraints: No hardcoding specific coordinates.

```
#include <iostream>
using namespace std;
// Function prototypes for the four line segments
void drawSlope1(int rows, int columns); // First line segment
void drawSlope2(int rows, int columns); // Second line segment
void drawSlope3(int rows, int columns); // Third line segment
void drawSlope4(int rows, int columns); // Fourth line segment

int main() {
    const int rows = 50; // Height of the grid
    const int columns = 50; // Width of the grid
    // Draw each slope separately
    drawSlope1(rows, columns);
    drawSlope2(rows, columns);
    drawSlope3(rows, columns);
    drawSlope4(rows, columns);
    return 0;
}
// First line segment [5 marks]
void drawSlope1(int rows, int columns) {
    // Write your logic here
}
// Second line segment [5 marks]
```

```
void drawSlope2(int rows, int columns) {
    // Write your logic here
}
// Third line segment [5 marks]
void drawSlope3(int rows, int columns) {
    // Write your logic here
}
// Fourth line segment [5 marks]
void drawSlope4(int rows, int columns) {
    // Write your logic here
}
```

2.6 Question 7 [10 Marks]

You are tasked to simulate the movement of two square robots (R1 and R2), each represented as a 4x4 block of * symbols within a grid. Each robot moves randomly in one of four directions (up, down, left, right) at each step. If a robot collides with the grid boundary, it reverses direction. If the two robots collide, they both reverse their paths.

The code for the above simulation is given. Your task is to complete the logic for:

1. **Robot movement logic:** Update the robot's position based on its direction (dx, dy).
2. **Wall collisions:** Detect if any part of a robot collides with the grid boundary.
3. **Robot collisions:** Detect if the two robots overlap.

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

const int GRID_SIZE = 20; // Grid dimensions
const int ROBOT_SIZE = 4; // Size of the robots (4x4)

// Function prototypes
void initializePositions(int &x1, int &y1, int &x2, int &y2);
void displayGrid(int x1, int y1, int x2, int y2);

// TO BE IMPLEMENTED:
void moveRobot(int &x, int &y, int dx, int dy);
bool checkWallCollision(int x, int y);
bool checkRobotCollision(int x1, int y1, int x2, int y2);

int main() {
    srand(time(0));
    int x1, y1, x2, y2; // Top-left positions of R1 and R2
    int dx1 = 1, dy1 = 0; // Direction of R1
    int dx2 = 0, dy2 = 1; // Direction of R2

    initializePositions(x1, y1, x2, y2);

    while (true) {
        system("clear"); // Clears the screen
        displayGrid(x1, y1, x2, y2);

        // Move R1
        moveRobot(x1, y1, dx1, dy1);
```

```
        if (checkWallCollision(x1, y1)) {
            dx1 = -dx1; dy1 = -dy1; // Reverse direction
        }

        // Move R2
        moveRobot(x2, y2, dx2, dy2);
        if (checkWallCollision(x2, y2)) {
            dx2 = -dx2; dy2 = -dy2; // Reverse direction
        }

        // Check robot collision
        if (checkRobotCollision(x1, y1, x2, y2)) {
            dx1 = -dx1; dy1 = -dy1;
            dx2 = -dx2; dy2 = -dy2;
        }

        // Pause to simulate animation
        for (int i = 0; i < 100000000; i++); // Simple delay
    }
    return 0;
}

// Function to initialize positions of the robots
void initializePositions(int &x1, int &y1, int &x2, int &y2) {
    x1 = rand() % (GRID_SIZE - ROBOT_SIZE);
    y1 = rand() % (GRID_SIZE - ROBOT_SIZE);
    do {
        x2 = rand() % (GRID_SIZE - ROBOT_SIZE);
        y2 = rand() % (GRID_SIZE - ROBOT_SIZE);
    } while ((x1 < x2 + ROBOT_SIZE && x1 + ROBOT_SIZE > x2) &&
             (y1 < y2 + ROBOT_SIZE && y1 + ROBOT_SIZE > y2)); // Ensure no initial overlap
}

// Function to display the grid
void displayGrid(int x1, int y1, int x2, int y2) {
    for (int i = 0; i < GRID_SIZE; i++) {
        for (int j = 0; j < GRID_SIZE; j++) {
            if (i >= x1 && i < x1 + ROBOT_SIZE && j >= y1 && j < y1 + ROBOT_SIZE) {
                cout << "*";
            } else if (i >= x2 && i < x2 + ROBOT_SIZE && j >= y2 && j < y2 + ROBOT_SIZE) {
                cout << "*";
            } else {
                cout << ".";
            }
        }
        cout << endl;
    }
}

// --- STUDENT CODE GOES HERE ---

// Write move robot logic here [10 marks]
void moveRobot(int &x, int &y, int dx, int dy) {
    // ...
```

```
}

// Write wall collision logic here [10 marks]
bool checkWallCollision(int x, int y) {
    // ...
}

// Write Robot collision logic here [10 marks]
bool checkRobotCollision(int x1, int y1, int x2, int y2) {
    // ...
}
```

2.7 Question 8 [20 Marks]

Below is a program to analyze a sentence entered by the user. The program performs the following tasks:

1. Break the sentence into individual words and store them in a dynamically created 2D array using double pointers.
2. Find and print the longest word in the sentence.
3. Calculate and print the total length of the sentence, excluding spaces.
4. Delete all dynamically allocated memory to avoid memory leaks.

Complete the functions: `breakSentenceIntoWords`, `findLongestWord`, `calculateTotalLength`, `deleteWordArray`.

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

// Function Prototypes
void breakSentenceIntoWords(char* sentence, char**& words, int& wordCount);
char* findLongestWord(char** words, int wordCount);
int calculateTotalLength(char** words, int wordCount);
void deleteWordArray(char** words, int wordCount);

int main() {
    const int MAX_LENGTH = 256;
    char sentence[MAX_LENGTH];
    // Input: Read a sentence from the user
    cout << "Enter a sentence (max 256 characters): ";
    cin.getline(sentence, MAX_LENGTH);

    // Variables for dynamically storing words
    char** words = nullptr;
    int wordCount = 0;

    // Task 1: Break sentence into words
    breakSentenceIntoWords(sentence, words, wordCount);

    // Task 2: Find and print the longest word
    cout << findLongestWord(words, wordCount);

    // Task 3: Calculate and print the total length (excluding spaces)
    int totalLength = calculateTotalLength(words, wordCount);
    cout << "Total length of the sentence (excluding spaces): " << totalLength << endl;

    // Task 4: Delete dynamically allocated memory
```

```
    deleteWordArray(words, wordCount);
    return 0;
}

// --- STUDENT CODE GOES HERE ---

// Task 1: Break the sentence into words and store in a dynamically created 2D array [15 marks]
void breakSentenceIntoWords(char* sentence, char**& words, int& wordCount) {
    // ...
}

// Function to find the longest word in the array
char* findLongestWord(char** words, int wordCount) {
    // ...
}

// Function to calculate the total length of the sentence (excluding spaces)
int calculateTotalLength(char** words, int wordCount) {
    // ...
}

// Task 4: Delete dynamically allocated memory for the words array [5 marks]
void deleteWordArray(char** words, int wordCount) {
    // ...
}
```

2.8 Question 9 [10 Marks]

You are given a partially completed program that performs several mathematical operations. Write Correct Function Prototypes for the given functions. For each function decide whether arguments should be passed by value, reference, or pointer, and the return type.

1. **Function 1: calculateSum** - The result should directly update a provided variable.
2. **Function 2: findMinMax** - Finds the smallest and largest values in an array and updates them.
3. **Function 3: conditionalSwap** - The swap should modify the original variables directly.

```
// Function Definitions (Prototypes Missing - Fill the blanks)

// Function 1: The result should directly update a provided variable. [2.5 marks]
void calculateSum(_____________) { // Student to fill prototype args
    int sum = 0; // Ensure sum starts at 0
    for (int i = 0; i < size; i++) {
        sum += arr[i];
    }
}

// Function 2: Finds the smallest and largest values in an array and updates them. [2.5 marks]
void findMinMax(_____________) { // Student to fill prototype args
    int smallest = arr[0];
    int largest = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] < smallest) smallest = arr[i];
        if (arr[i] > largest) largest = arr[i];
    }
}
```

```
// Function 3: The swap should modify the original variables directly. [2.5 marks]
void conditionalSwap(_____) { // Student to fill prototype args
    if (x > y) {
        int temp = x;
        x = y;
        y = temp;
    }
}
```