National University of Computer and Emerging Sciences

Islamabad Campus

Programming Fundamentals (CS1002)

Course Instructor(s):
Mr. Bilal Khalid Dar
Section(s): SE-A, SE-B, SE-C

Final Examination

Total Time (Hrs): 3
Total Marks: 170
Total Questions: 09

Date: Jan 6, 2025

______________ ______________ __________________________

Roll No **Course Section** **Student Signature**

Do not write below this line.

PART - A

Attempt all the questions of part A on the Question paper.

Instructions:

- **Attempts all questions of Part A on the Question paper. Write all answers in the provided space.**
- **Cutting and Overwriting will result in zero marks**

[CLO 1: Demonstrate the basic concepts of programming]

Q1: **[10 marks]**

In C++, variable names must follow specific rules. Below is a list of variable names. Identify which of them are valid and which are invalid in C++.

	Variable	**Valid/Invalid**
1	int myVar;	Valid
2	float 2ndValue;	Invalid
3	char _temp;	Valid
4	double total-amount;	Invalid
5	bool isValid;	Valid
6	int $rate;	Invalid
7	float area1;	Valid
8	double size@home;	Invalid
9	int _123abc;	Valid
10	unsigned int number_1;	Valid

Either 1 or zero. Cutting and overwriting zero marks

Q2: **[5 marks]**

Conver the following mathematical expression into C++ expression. You can use math.h functions. Assume y is of type double

$$y = \frac{(a+b)^3 - (c \cdot d)}{\sqrt{e + |f - g|}}$$

Solution:

double y = (pow(a + b, 3) - (c * d)) / sqrt(e + abs(f - g));

5 or zero. Only equation was required

Q3: **[5 marks]**

What is the output of the following C++ expressions.

	Expression	**Output**
Example	**int a = 5 + 5;**	**10**
1	float result = 5 / 2.0;	2.5
2	int result = 7.8 + 2.5;	10
3	float result = 5.5 % 2;	Error
4	float result = (5 / 2) * 1.5;	3.0
5	float result = 5.0 * 2.0 / (4.0 * 2.0);	1.25

Either 1 or zero. Cutting and overwriting zero marks

Q4: **[5x5 = 25 marks]**

Identify the exact errors in the following programs (The line or phase that can cause the syntax, runtime or logical error).

	Program	**Error**
E.g.	1. #include <iostream> 2. using namespace std; 3. int main() 4. { 5. cout << "Hello World" 6. }	**Line 5 has not semi colon and will result in syntax error.**
1	1. #include <iostream> 2. using namespace std; 3. int main() 4. { 5. int n = 5; 6. if (n = 0) 7. cout << "n is zero" << ".\n"; 8. else 9. cout << "n is not zero" << ".\n"; 10. cout << "The square of n is " << n * n << ".\n"; 11. return 0; 12. }	**Line 6** **Logical n = 0 should be n == 0**
2	1. #include <iostream> 2. using namespace std; 3. 4. int main() { 5. int* ptr; 6. *ptr = 10; 7. cout << "Value: " << *ptr << endl; 8. return 0; 9. }	**Line 5** ptr is uninitialized, leading to undefined behavior when dereferenced. Logical Error

	10.	
3	1. #include <iostream> 2. using namespace std; 3. 4. int main() { 5. int arr[5] = {10, 20, 30, 40, 50}; 6. int* ptr = arr; 7. cout << "Value: " << *(ptr + 5) << endl; 8. return 0; 9. }	**Line 7 Exceeding array size** **Out of bound error**

4	1. #include <iostream>
	2. using namespace std;
	3. void increment(int* p) {
	4. p++;
	5. }
	6. int main() {
	7. int x = 5;
	8. increment(&x);
	9. cout << x << endl;
	10. return 0;
	11. }
		Line 4 issue , increment
		Incorrect address increment.

5	1. #include <iostream>
	2. using namespace std;
	3. int getSum(int a, int b = 0, int c) {
	4. return a + b + c;
	5. }
	6. int main() {
	7. int num1 = 5, num2 = 10, num3 = 15;
	8. cout << "Sum of " << num1 << " and " << num2 << " is: " << getSum(num1, num2) << endl;
	9. cout << "Sum of " << num1 << ", " << num2 << ", and " << num3 << " is: " << getSum(num1, num2, num3) << endl;
	10. cout << "Sum of " << num1 << " is: " << getSum(num1) << endl;
	11. return 0;
	12. }
		Issue with line 3, incorrect default argument
		Syntax error

Q5: **[2x5 = 10 marks]**

Define a 2D integer array in C++ with the following requirements

1. Take space in stack

```
int main() {
    const int rows = 3, cols = 4; // Example dimensions
    int arr[rows][cols]; // 2D array allocated on the stack
    return 0;
}
```

2. Take space in Heap

```
int main() {
    const int rows = 3, cols = 4; // Example dimensions
    int** arr = new int*[rows]; // Array of pointers (allocated on heap)
    for (int i = 0; i < rows; i++) {
        arr[i] = new int[cols]; // Allocate each row on the heap
    }
```

Q6: **[15x5 = 75 marks]**

Write the output of the following programs. Assume all the header files and there are no syntax errors. Write the exact output of the programs below. Indicate a blank space in the output by writing the symbol ☐. Indicate a blank line in the output by writing blank line __

	Program	**Output**

1	`#include <iostream>`
	`using namespace std;`
	
	`int main() {`
	`    int arr[2][2] = {{10, 20}, {30, 40}};`
	`    int* p = (int*)arr;`
	`    cout << *(p + 1) << " " << *(p + 3) << " " << *p << endl;`
	`    return 0;`
	`}`
		20 40 10

2	`#include <iostream>`
	`using namespace std;`
	
	`int main() {`
	`    int arr[3][2] = {{1, 2}, {3, 4}, {5, 6}};`
	`    int(*p)[2] = arr;`
	`    cout << p[0][1] << " " << p[1][0] << " " << p[2][1] << endl;`
	`    return 0;`
	`}`
		2 3 6

3	#include <iostream>
	using namespace std;
	
	int main() {
	    int a = 10, b = 20;
	    int* p = &a;
	    int* const* dp = &p;
	    p = &b;
	    cout << **dp << endl;
	    return 0;
	}
		20

4	#include <iostream>
	using namespace std;
	
	int main() {
	    int a = 1, b = 2, c = 3;
	    int* arr[] = {&a, &b, &c};
	    cout << *arr[0] << " " << *arr[1] << " " << *arr[2] << endl;
	    return 0;
	}
		1 2 3

5	#include <iostream>
	using namespace std;
	
	int main() {
	    int arr[5] = {2, 4, 6, 8, 10};
	    int* p = arr + 2;
	    cout << *p << " " << *(p - 2) << " " << *(p + 2) << endl;
	    return 0;
	}
		6 2 10

6	#include <iostream>
	using namespace std;
	
	int main() {
	    int arr[] = {5, 10, 15};
	    int* p = arr;
	    int** dp = &p;
	    cout << **dp << " " << *(*dp + 1) << " " << *p << endl;
	    return 0;
	}
		5 10 5

7	#include <iostream>
	using namespace std;
	int main()
	{
	    char ch;
	      char title[] = "Titanic";
	     ch = title[1];
	     title[3] = ch;
		Titinic
		
		i

<table>
<tr><td></td><td><pre><code>   cout << title << endl;
     cout << ch << endl;

   return 0;
}</code></pre></td><td></td></tr>
<tr><td><b>8</b></td><td><pre><code>#include <iostream>
using namespace std;
int main()
{
   int i = 5, j = 6, k = 7, n = 3;
   cout << i + j * k - k % n << endl;
   cout << i / n << endl;

   return 0;
}</code></pre></td><td><b>46</b>
<b>1</b></td></tr>
<tr><td><b>9</b></td><td><pre><code>#include <iostream>
using namespace std;
int main()
{
   int n = 0;
   cout << (n = 4) << endl;
   cout << (n == 4) << endl;
   cout << (n > 3) << endl;
   cout << (n < 4) << endl;
   cout << (n = 0) << endl;
   return 0;
}</code></pre></td><td><b>4</b>
<b>1</b>
<b>1</b>
<b>0</b>
<b>0</b></td></tr>
<tr><td><b>10</b></td><td><pre><code>#include <iostream>
using namespace std;
void modify(int **ptr) {
    **ptr = 100;
}
int main() {
    int x = 50;
    int *p = &amp;x;
    modify(&p);
    cout << x << endl;
    return 0;
}</code></pre></td><td><b>100</b></td></tr>
<tr><td><b>11</b></td><td><pre><code>#include <iostream>
using namespace std;

int main() {
    int *arr = new int[4];
    for (int i = 0; i < 4; i++) {
        arr[i] = (i + 1) * 10;
    }
    for (int i = 0; i < 4; i++) {
        cout << arr[i] << " ";
    }
    delete[] arr;
    return 0;
}</code></pre></td><td><b>10 20 30 40</b></td></tr>
</table>

<table>
<tr>
<td><b>12</b></td>
<td>

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

int main() {
    int arr[3][3] = {{1, 2, 3}, {4, 5, 6}, {7,
8, 9}};
    int (*ptr)[3] = arr;
    cout << *(*(ptr + 1) + 1) << " ";
    cout << *(*(ptr + 2) + 0) << endl;
    return 0;
}
```

</td>
<td><b>5 7</b></td>
</tr>
<tr>
<td><b>13</b></td>
<td>

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

int main() {
    int arr[] = {3, 6, 9};
    int *ptr = arr;
    int **dptr = &ptr;
    cout << **dptr << " ";
    (*dptr)++;
    cout << **dptr << " ";
    ptr++;
    cout << **dptr << endl;
    return 0;
}
```

</td>
<td><b>3 6 9</b></td>
</tr>
<tr>
<td><b>14</b></td>
<td>

```
#include <iostream>
using namespace std;
void fun() {
    static int c = 0;
    c++;
    cout << c << endl;
}
int main() {
    fun();      fun();      fun();      fun();
    return 0;
}
```

</td>
<td>

```
1
2
3
4
```

</td>
</tr>
<tr>
<td><b>15</b></td>
<td>

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

int main() {
    int r = 4;

    for (int i = 1; i <= r; ++i) {
        for (int j = 1; j <= r - i; ++j) {
            cout << " ";
        }
        for (int j = 1; j <= (2 * i - 1); ++j) {
            cout << "*";
        }
        cout << endl;
    }
    return 0;
}
```

</td>
<td>

```
   *
  ***
 *****
*******
```

</td>
</tr>
</table>

PART - B

Attempt all the questions of part B on the Answer Sheet.

[CLO 2: Apply algorithmic solutions related to the degree program to recent related problems]

Q7: **[10 marks]**

Write a function named "reverse" that takes as its arguments the following:
(1) an array of floating-point values.
(2) an integer that tells how many floating-point values are in the array.

The function must reverse the order of the values in the array. Thus, for example, if the array that's passed to the function looks like this:

0	**1**	**2**	**3**	**4**
5.8	2.6	9.0	3.4	7.1

the original array will have been modified so that it looks like this:

0	**1**	**2**	**3**	**4**
7.1	3.4	9.0	2.6	5.8

- The function should not return any value.
- Please note that you cannot define any new array for solving this problem.
- **Note: Add comments to explain your logic as well. Your solution should be generic.**

```
void reverse(float arr[], int size) {
   int start = 0;           // Initialize the start index
   int end = size - 1;      // Initialize the end index

   // Loop until the start index is less than the end index
   while (start < end) {
      // Swap the elements at start and end indices
      float temp = arr[start];
      arr[start] = arr[end];
      arr[end] = temp;

      // Move the start index forward and the end index backward
      start++;
      end--;
   }
}
```

Q8: **[15 marks]**

Write a function to find the longest word in the string. The function should return the longest word. You cannot use any build in string function other than length.

Example
Input: My name is Bilal Khalid Dar
Longest Word in the String: Khalid

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

// Function to find the longest word in a string
string findLongestWord(const string& str) {
    int maxLength = 0;        // Variable to store the length of the longest word
    int currentLength = 0;    // Variable to store the length of the current word
    string longestWord = "";  // Variable to store the longest word
    string currentWord = "";  // Variable to store the current word

    // Loop through each character in the string
    for (int i = 0; i < str.length(); i++) {
        if (str[i] != ' ') {
            // If the character is not a space, it's part of a word
            currentWord += str[i];  // Add the character to the current word
            currentLength++;         // Increase the current word's length
        } else {
            // If the character is a space, check if the current word is the longest
            if (currentLength > maxLength) {
                maxLength = currentLength;  // Update the maximum length
                longestWord = currentWord;  // Update the longest word
            }
            // Reset the current word and its length
            currentWord = "";
            currentLength = 0;
        }
    }

    // Final check in case the longest word is the last word in the string
    if (currentLength > maxLength) {
        longestWord = currentWord;
    }

    return longestWord;  // Return the longest word
}

int main() {
    string str = "My name is Bilal Khalid Dar";  // Example input string
    string longest = findLongestWord(str);  // Call the function to find the longest
word
    cout << "Longest Word in the String: " << longest << endl;  // Output the
result
    return 0;
}
```

Q9: **[15 marks]**

Write a C++ program to find the union of two arrays. The union of two arrays includes all unique elements from both arrays. Implement the solution using a function and pointers. The function should take two arrays and their respective sizes as input and return a dynamically allocated array containing the union.

Example Input:

Array1: {1, 2, 3, 4}

Array2: {3, 4, 5, 6}

Output: Union: {1, 2, 3, 4, 5, 6} (Hint DMA)

- Use pointers to traverse and manipulate arrays.
- Ensure no duplicate elements appear in the result.
- Display the size and output of the resultant union array.

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

// Function to find the union of two arrays
int* findUnion(int* arr1, int size1, int* arr2, int size2, int& unionSize)
{
    // Allocate dynamic memory for the union array (maximum size will be
size1 + size2)
    int* unionArr = new int[size1 + size2];
    int index = 0;  // To keep track of the current position in the union
array

    // Add elements from arr1 to the union array
    for (int i = 0; i < size1; ++i) {
        unionArr[index++] = arr1[i];
    }

    // Add elements from arr2 if not already in the union array
    for (int i = 0; i < size2; ++i) {
        bool isDuplicate = false;
        for (int j = 0; j < size1; ++j) {
            if (arr2[i] == arr1[j]) {
                isDuplicate = true;  // Element is already in arr1, so skip
it
                break;
            }
        }
        // If no duplicate is found, add it to the union array
        if (!isDuplicate) {
            unionArr[index++] = arr2[i];
        }
    }
    // Update the size of the union array
    unionSize = index;

    return unionArr;  // Return the dynamically allocated union array
}

int main() {
    // Sample input arrays
    int arr1[] = {1, 2, 3, 4};
    int arr2[] = {3, 4, 5, 6};
    int size1 = sizeof(arr1) / sizeof(arr1[0]);  // Calculate the size of
arr1
    int size2 = sizeof(arr2) / sizeof(arr2[0]);  // Calculate the size of
arr2

    // Call the function to get the union of the arrays
    int unionSize = 0;
    int* unionArr = findUnion(arr1, size1, arr2, size2, unionSize);

    // Display the resultant union array
    cout << "Union: {";
    for (int i = 0; i < unionSize; ++i) {
        cout << unionArr[i];
        if (i < unionSize - 1) {
            cout << ", ";
        }
    }
    cout << "}" << endl;

    // Display the size of the union array
    cout << "Size of Union: " << unionSize << endl;

    // Free the dynamically allocated memory
    delete[] unionArr;

    return 0;
}
```