

**CAT’s Marking Guide**

ACADEMIC YEAR: **2023-2024**

LEVEL: **5 (RCA YEAR 2)**

TERM: **1**

COURSE TITLE: **DATA STRUCTURES AND ALGORITHMS WITH C++**

MODULE CODE: **SPEDA402**

NUMBER OF TEACHING HOURS/WEEK: **5**

DATE: **16/11/2023**

DURATION: **180 Min**

MAXIMUM MARKS: **50**

**INSTRUCTIONS:**

1. This examination paper comprises **TWO** sections.
2. Answer **ALL QUESTIONS** from section **A.**
3. Answer **TWO (2)** questions from section **B.**

**Examiners:** HABANABASHAKA Jean Damascene & HATANGIMBABAZI Hilaire **Section A: Answer ALL Questions /30 Points**

1. Among the features of C++, it is said to be **Portable, Object Oriented, Case sensitive** and **Type checking.** Explain these **(4 marks)**

**Answer:**

* C++ programming language is **portable**; this means that programs written in C++ are portable across multiple hardware and software platforms.
* Being **object-oriented** programming language means that the design goal of C++ is to support object- oriented programming. Instead of using functions that access global variables, both data and variables are encapsulated into an object.
* C++ is **case sensitive** whichmeans that uppercase and lowercase letters are treated as different characters. For example, in C++ a and A are different.
* It is also **type checking** because it provides the rules and mechanism for checking data types before execution starts.
  1. What do you understand by the term **namespace** in C++? **(1 mark)**

**Answer:**

A namespace is a declarative region that provides a scope to the identifiers (the names of types, functions, variables, etc) inside it. **1 mark** They are used to organize code into logical groups and to prevent name collisions that can occur especially when your code base includes multiple libraries. **1 mark**

* 1. What is the role of **using namespace std;** statement in C++ programs? **(1 mark)**

**Answer:**

It means that we can use names of objects and variables from the standard namespace (std). **1 mark**

* 1. With an example, give another alternative in the case the statement in b) is not used in a C++ program. **(1 mark)**

**Answer:**

The alternative in the case the statement **using namespace std;** is not used is **std::identifier\_defined\_in\_std\_namespace; 0.5 marks**

**Example: std::cin>>varName; 0.5 marks**

1. Give different ways of reading/inputting strings in C++ **(2 marks)**

**Answer:**

In C++ the string is read using **cin1 mark** statement or **getline()1 mark** function.

However, **cin** considers a space (whitespace, tabs, etc) as a terminating character, which means that it can only store a single word (even if you type many words) **1 mark** whereas **getline()** reads the sting as it is **1 mark**

1. Using an example, explain how the string concatenation is done in C++ **(1 mark)**

**Answer:**

* In C++ the string concatenation uses the **strcat()** predefined function. It takes two strings to be concatenated as arguments. **0.5 mark**
* Using + operator **0.5 mark**

Example: **strcat(“hello ”,”word”)** or **“hello ”+”word”** bothresult **hello word 0.5 mark**

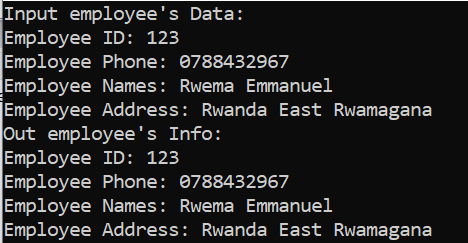
1. Write a C++ program to receive employee’s information from the user and output it on console.

The program has 4 string variables ***employeeId, employeePhone, employeeNames***(to store all employee’s names (first, middle and last names) at the same time)and ***employeeAddress*** (To store all employee’s Country, Province and District at the same time)***.* (4 marks)**

**Answer: NB: Many programs can be written differently to solve this**

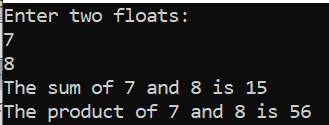


**Sample output (Not mandatory)**



1. In a C++ program, define a namespace with the following specifications:

* The namespace name is **artimeticOperations**
* **artimeticOperations** has 2 functions **add()** and **mult()**
* Both **add()** and **mult()** receive two float parameters and return their sum and product respectively.
* The program has two global variables **v1** and **v2**
* The values of **v1** and **v2** are given by the user at run time in main function and passed to **add()** and **mult()** functions**.**
* The output should be like indicated below:



**(4 marks)**

**Answer: NB: Many programs can be written differently to solve this**

A computer screen shot of a computer code

Description automatically generated

1. Using examples, classify constants in C++. **(2 marks)**

**Answer:**

In C++, are classified into **literal constants** and **symbolic constants. 1 mark**

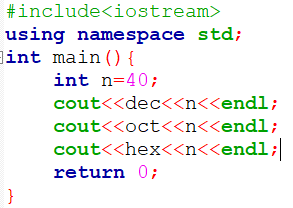
**Examples:**

* **literal constant:** a-**7 0.5 marks**
* **Symbolic Constants:**

**#define PI 3.14 0.5 marks**

**const int I = 5 0.5 marks**

1. What is the output of the following C++ program: **(3 marks)**



**40**

**50**

**28**

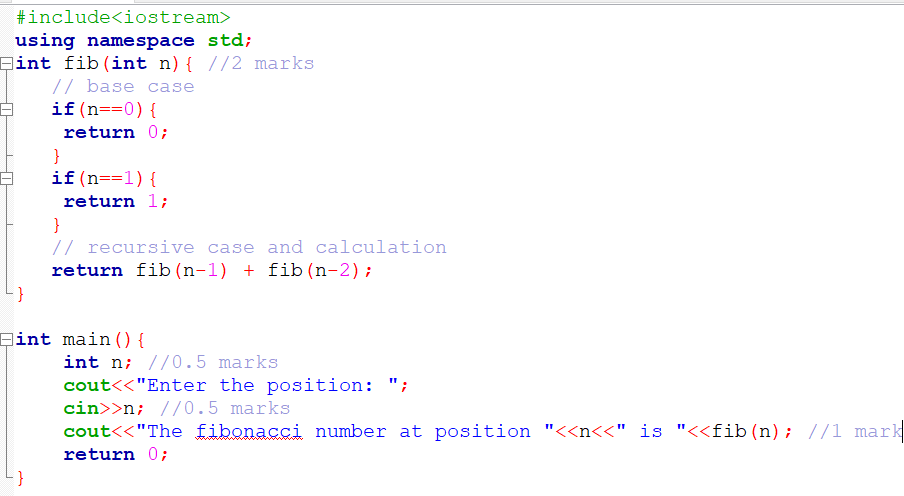
1. What is the result of the following C++ statements **(2 marks)**
   1. **cout<<ceil(8.2);**
   2. **cout<<fmod(2.7,1.2);**
   3. **cout<<isdigit(‘5’)**
   4. **cout<< strcat(“x”,”y”);**

**Answer:**

* 1. **9 0.5 marks**
  2. **0.3 0.5 marks**
  3. **1 0.5 marks**
  4. **xy 0.5 marks**

1. Using the recursive algorithms, define a function in a C++ program to find the Fibonacci number at position n. The first two numbers in the Fibonacci series are 0 and 1 and for any other position, the number is equal to the sum of the preceding two numbers. **(3 marks)**

**Answer**



1. Differentiate time from space complexities for algorithms **(1 mark)**

**Answer:**

Time complexity of an algorithm is a function that describes **how long an algorithm takes in terms of the quantity of input (0.5 marks)** it receives whereas the space complexity is a function that describes **how much memory (space) an algorithm requires to the quantity of input (0.5 marks)** to the method.

1. What is the difference between an array and vector in C++? **(1 mark)**

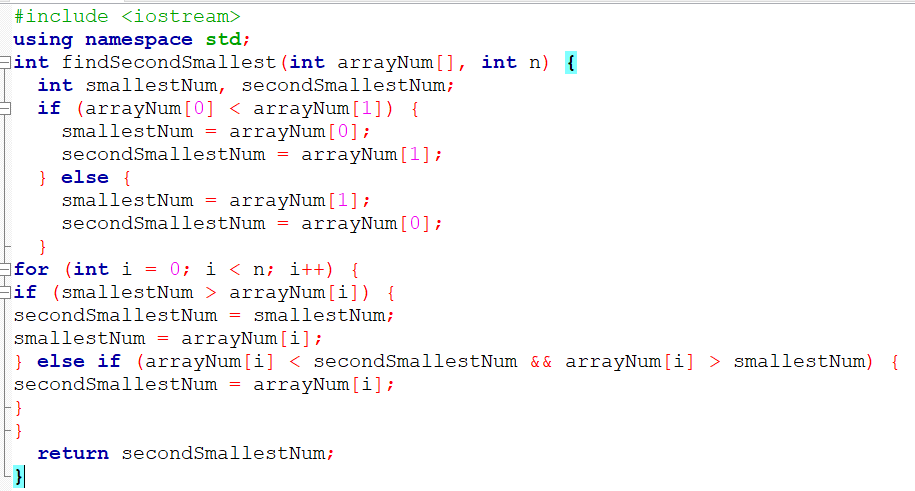
**Answer:**

Any correct difference **1 mark**

**Section B: Answer ONLY TWO (2) Questions /20 Points**

1. Write a C++ program implementing an algorithm to find the second smallest elements in a given array of integers. What is the time complexity of the algorithm you implemented? **(10 marks)**

**Answer:**



A computer code with text

Description automatically generated

**Sample output: (Not part of the answer)**

A black screen with white text

Description automatically generated

The time complexity will depend on every one’s implementation. In this case, the algorithm is summarized like this:

1. Declare empty array and its size
2. Input the size of the array.
3. Iterate through the array.

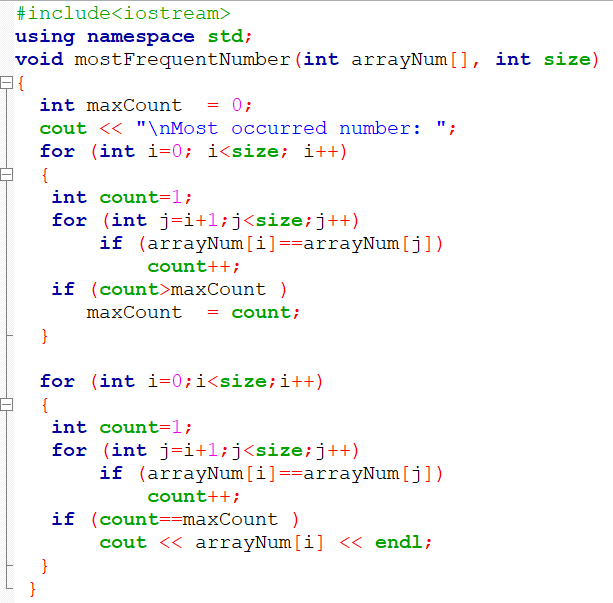
* Input elements of the array

1. Initialize two variables, **smallestNumber** and **secondSmallestNumber**, depending on the first two elements in the array.
2. Iterate through the array.

* If the current element is smaller than **smallestNumber**, update both **smallestNumber** and **secondSmallestNumber**
* If the current element is greater than **smallestNumber** but smaller than **secondSmallestNumber**, update **secondSmallestNumber**.

1. Return **secondSmallestNumber**.

The time complexity of this approach is **O(n)**, where 'n' is the number of elements in the array.

1. Write a C++ program to implement an algorithm to find the most frequent element in an array of integers. What is the time complexity of implemented algorithm? **(10 marks)**

**Answer**

A computer code with text

Description automatically generated

**Sample output(Not part of the answer)**

A screenshot of a computer

Description automatically generated

Like in question 14, the time complexity will depend on the approach used.

In this case, the algorithm is the one which compares each element with every other element in the array. This implies that the Time complexity is **O(n^2)** in the worst case, where n is the number of elements in the array.

Another case can be using sorting in which the array is sorted and then traversed to find the most frequent element. In this case the time complexity will be **O(n log n)**

1. Fill in empty cells of the table below concerning vector functions: **(10 marks)**

**Function:** Description

**begin():**-----------------------

**------------:** Returns an iterator pointing to the theoretical element that follows the last element in the vector.

**---------------**: Returns a reverse iterator pointing to the last element in the vector (reverse beginning). It moves from last to first element.

**rend():**------------------

**----------------**: Returns a constant iterator pointing to the first element in the vector.

**----------------**: Returns a constant iterator pointing to the theoretical element that follows the last element in the vector.

**crbegin()**: ---------------------

**--------------------**: Returns a const reverse iterator pointing to the theoretical element preceding the first element in the vector (considered as reverse end).

**size()**: Returns the number of elements in the vector

**------------------**: Returns the maximum number of elements in the vector

**SMILE IS ELECTRIC AGTuhCapacity():** -------------------------

**Answer**

**Function:** Description

**begin():** returns an iterator pointing to the first element in the vector **1 mark**

**end() 1 mark:** Returns an iterator pointing to the theoretical element that follows the last element in the vector.

**rbegin() 1 mark**: Returns a reverse iterator pointing to the last element in the vector (reverse beginning). It moves from last to first element.

**rend():** Returns a reverse iterator pointing to the theoretical element preceding the first element in the vector (considered as reverse end) **1 mark**

**cbegin()1 mark**: Returns a constant iterator pointing to the first element in the vector.

**cend()**:**1 mark** Returns a constant iterator pointing to the theoretical element that follows the last element in the vector.

**crbegin()**: Returns a const reverse iterator pointing to the last element in the vector (reverse beginning). It moves from last to first element. **1 mark**

**crend()1 mark**: Returns a const reverse iterator pointing to the theoretical element preceding the first element in the vector (considered as reverse end).

**size()**: Returns the number of elements in the vector

**max\_size()1 mark**: Returns the maximum number of elements in the vector

**Capacity():** Returns the size of the storage space currently allocated to the vector expressed as the number of elements **1 mark**

**End**

**///////////////////////////////////////////////////////////**

**Programs for Questions 13 and 14 in editable way**

**///////////////////////////////////////////////////////////**

**//// Q13:**

**#include <iostream>**

**using namespace std;**

**int findSecondSmallest(int arrayNum[], int n) {**

**int smallestNum, secondSmallestNum;**

**if (arrayNum[0] < arrayNum[1]) {**

**smallestNum = arrayNum[0];**

**secondSmallestNum = arrayNum[1];**

**} else {**

**smallestNum = arrayNum[1];**

**secondSmallestNum = arrayNum[0];**

**}**

**for (int i = 0; i < n; i++) {**

**if (smallestNum > arrayNum[i]) {**

**secondSmallestNum = smallestNum;**

**smallestNum = arrayNum[i];**

**} else if (arrayNum[i] < secondSmallestNum && arrayNum[i] > smallestNum) {**

**secondSmallestNum = arrayNum[i];**

**}**

**}**

**return secondSmallestNum;**

**}**

**int main() {**

**int n;**

**cout<<"Enter the number of elements: ";**

**cin>>n;**

**int arrayNum[n]={};**

**cout<<"\n Enter "<<n<<" elements of array:\n";**

**for(int i=0;i<n;i++){**

**cout<<"array\_num["<<i<<"]:";**

**cin>>arrayNum[i];**

**}**

**int s = sizeof(arrayNum) / sizeof(arrayNum[0]);**

**cout << "Original array: ";**

**for (int i=0; i < s; i++)**

**cout << arrayNum[i] <<" ";**

**int secondSmallestNum = findSecondSmallest(arrayNum, n);**

**cout<<"\nSecond smallest number: "<<secondSmallestNum;**

**return 0;**

**}**

**/////Q14**

**//////////////**

**#include<iostream>**

**using namespace std;**

**void mostFrequentNumber(int arrayNum[], int size)**

**{**

**int maxCount = 0;**

**cout << "\nMost occurred number: ";**

**for (int i=0; i<size; i++)**

**{**

**int count=1;**

**for (int j=i+1;j<size;j++)**

**if (arrayNum[i]==arrayNum[j])**

**count++;**

**if (count>maxCount )**

**maxCount = count;**

**}**

**for (int i=0;i<size;i++)**

**{**

**int count=1;**

**for (int j=i+1;j<size;j++)**

**if (arrayNum[i]==arrayNum[j])**

**count++;**

**if (count==maxCount )**

**cout << arrayNum[i] << endl;**

**}**

**}**

**int main()**

**{**

**int n;**

**cout<<"Enter the number of elements: ";**

**cin>>n;**

**int arrayNum[n]={};**

**cout<<"\n Enter "<<n<<" elements of array:\n";**

**for(int i=0;i<n;i++){**

**cout<<"array\_num["<<i<<"]:";**

**cin>>arrayNum[i];**

**}**

**int s = sizeof(arrayNum)/sizeof(arrayNum[0]);**

**cout << "Original array: ";**

**for (int i=0; i < s; i++)**

**cout << arrayNum[i] <<" ";**

**mostFrequentNumber(arrayNum, n);**

**}**