| 09:00–11:00 hrs | 14 / June / 2024 | Exams Hall | EXAMS OFFICE USE ONLY |
|---|---|---|---|

University of the Witwatersrand, Johannesburg

| | |
|---|---|
| Course or topic No(s) | COMS1018A, COMS1022A |
| Course or topic name(s) Paper Number & title | Introduction to Algorithms and Programming |
| Examination to be held during the month(s) of | June 2024 |
| Year of study | |
| Degrees/Diplomas for which this course is prescribed | |
| Faculties presenting candidates | Science |
| Internal examiner(s) | Dr Steven James<br>x-76157 |
| External examiner(s) | Pula Rammoko/Dr Linda Marshall (UP) |
| Special materials | None |
| Time allowance | 2 Hours |
| Instructions to candidates | 50 Marks available. 50 marks = 100%.<br>Answer all questions. This is a closed-book exam.<br>This exam consists of 12 pages, including this cover page. Write your student number on each page. |

Memo

Student Number: _____ Row: _____ Seat: _____

**For marking purposes only**

| | |
|---|---|
| Question 1 | |
| Question 2 | |
| Question 3 | |
| Question 4 | |
| Total | |

## Instructions

- Please write your student number on each page, including the front cover.
- Answer all questions in pen. **Do not write in pencil.**
- Please write your student number at the top of each page.
- This exam consists of 12 pages. Ensure that you are not missing any pages.
- This is a **closed-book** exam: you may not consult any written material or notes.
- You are allocated 3 hours to complete this exam.
- There are 4 questions and 50 marks available.
- Ensure your cellphone is switched off.

## Question 1 Short Questions [18 Marks]

1. What is the difference between the declaration and initialisation of a variable in C++? Provide an example of code to illustrate this difference. [2]

   Declaration: create a variable vs initialisation: create a variable with initial value

   Declaration: int x;

   Initialisation: int x = 2; 1mk for explanation, 1 for example

2. When a C++ program is run, the operating system must allocate memory in RAM for it. Which phase of the compilation cycle is responsible for this? [1]

   Load

3. In which part of memory are dynamically allocated variables stored? [1]

   Heap / free store

4. For each statement below, state whether it is true or false. [2]

   (a) Local, automatically allocated C++ variables are stored on the stack.

   True

   (b) Whitespace is semantically meaningful in C++ programs.

   False

   (c) If the value returned by the main function of a C++ program is 0, it means an error has occurred.

   False

   (d) If we wish to use smart pointers, then we must include the `<utility>` header file.

   False 0.5 each

5. Rewrite the declaration of the following C++11 array to create the equivalent dynamic array that is declared on the heap: `array<string, 10> words;` [1]

   string *words = new string[10];

6. Explain what a "memory leak" is and how this can occur. [2]

   When programmer forgets to release memory on the heap [1]

   Can happen when we forget to delete and then lose access to the pointer [1]

7. The value of the letter `g` in the ASCII table is 103. What is the value of `d`? [1]

   100

8. What header file should be included if we wish to use the built-in `sort` function? [1]

   algorithm

9. We are writing a program and wish to store the total amount of tax collected by the government in **cents**. What is the best built-in data type to use in this instance? Motivate your answer. [2]

   long (or long long, or long int) because it will be a massive roudn number, too big for int

   [give 1 part mark for int if it's well motivated]

   [1 mk for correct answer, 1 for motivation]

10. Write a single line of code that deletes a built-in array called `x`. [1]

    delete[] x;

11. Why might we prefer to pass a large object (such as a large `vector`) to a function by reference instead of by value? [1]

    Saves having to copy a large vector in memory (could save time and/or space)

12. How many elements does a vector declared using `vector<int> v{17};` contain? [1]

    1

13. What three elements make up a function's prototype in C++? [2]

    return type, function name, parameter list

    [2 for all three, 1 for two, 0.5 for one]

## Question 2 Code Flow [10 Marks]

1. Predict the output of the following code snippet. [1]

```
string s2 = "Hello";
string s1 = " World!";
cout << s2 << s1 << endl;
```

Hello World!

2. Predict the text that is displayed to the terminal when a user calls the function `f` in the code below. [2]

```
bool g(int x){
    if (x == 1){
        cout << "True" << endl;
        return false;
    }
    cout << "False" << endl;
    return true;
}

void f(){
    if (true && g(1)){
        cout << "True" << endl;
    }
    else{
        cout << "False" << endl;
    }
}
```

True [1]

False [1]

3. Predict the output of the following code. Assume that the code is executed on a machine where integers are represented using 32 bits and a single memory address is 64 bits wide. [3]

```
int x = 4;
int arr [16] = {0};
cout << sizeof(x) << " vs " << sizeof(&x) << " vs "
  << sizeof(arr)/sizeof(arr[0]) << endl;
```

4 vs 8 vs 16 [1 for each number]

4. Predict the output of the following code snippet. [2]

```
string a = "COMS";
int N = a.length();
for (int i = 0; i < N - 1; ++i){
    cout << i << ":";
    for (int j = i + 1; j < N; ++j){
        cout << " " << a[j];
    }
    cout << endl;
}
```

0: O M S
1: M S
2: S

2 for all correct

1 for wrong but some correct progression

5. Predict the output of the following code: [2]

```
int arr [4] = {3, 4};
int *b = &arr[1];
cout << *b << endl;
b++;
cout << *b << endl;
int *c = arr + 3;
cout << c - &arr[1] << endl;
cout << (arr == &arr[1] - 1) << endl;
```

4

0

2

0.5 each

1 or true

## Question 3 Debugging [12 Marks]

1. The following code reads five integers into a vector, doubles each element and then outputs the new elements on a single line. However, there are **three errors** in the code that will prevent it from compiling. For each error, identify it by line number, explain why it is an error and rewrite the line to correct it. [6]

```
1  #include <iostream>
2  #include <array>
3
4  using namespace std;
5
6  int main(){
7      int N = 5;
8      array<int, N> arr;
9      for (auto &element : arr){
10         cin << element;
11     }
12     multiply(arr, 2);
13     for (auto &element : arr){
14         cout << element << ' ';
15     }
16     cout << endl;
17     return 0;
18 }
19
20 void multiply(array<int, 5>& arr, int multiple){
21     for (auto &element : arr){
22         element *= multiple;
23     }
24 }
```

Line 7/8: Array size must be constant. Fix is either L7 (const int) or L8 (array<int, 5>)

L10: Arrows wrong way. Must be cin >> element

L12 or L20: Function used before declared. Either move multiply before main, or declare / put function prototype before main.

1 mk for each correct ID, 1 for correct fix

2. The following function accepts a single integer $N$. It then reads in $N$ integers from the user and outputs the minimum, maximum and mean values to the screen. However, the function occasionally outputs the wrong result. There are two issues *inside the function* that cause this incorrect behaviour. For each error, identify it by line number, explain why it is an error and rewrite the line to correct it. Note that the relevant header files, namespace, etc., are included but simply not shown here. [4]

```
1  void calculate(int N){
2      vector<int> nums(N);
3      sort(nums.begin(), nums.end());
4      for (int i = 0; i < N; ++i){
5            cin >> nums[i];
6      }
7      int min = nums[0];
8      int max = nums[N-1];
9      int sum = 0;
10     for (int x : nums){
11         sum += x;
12     }
13     double ave = sum / N;
14     cout << "The min is " << min << endl
15          << "The max is " << max << endl
16          << "The mean is " << ave << endl;
17 }
```

L3: Sort happens before numbers entered. Move sort to after loop on L6

L13: Integer division. Either sum or N should be double/float, should be cast/converted to float, or should be 1.0 * sum / N (or equivalent)

1 mk for each correct ID, 1 for correct fix

3. Your friend, Richard, is learning to program and has written the following C++ function to calculate the sum of a `vector` of integers:

```
int summation(vector<int> numbers){
    int sum = 0;
    for (int i = 0; i <= numbers.size(); i++){
        sum += numbers[i];
    }
    return sum;
}
```

However, he is confused by something strange! Most of the time, the function returns the correct result, but very occasionally, it returns completely the wrong value. He asks for your help, and you tell him the issue is that his loop runs one too many times: he should instead have written `i < numbers.size()`. Richard understands he has made an error, but he does not understand why the error causes his program to sometimes work, and sometimes fail. Explain what is happening by making reference to the underlying memory that is being accessed by his incorrect loop. Your answer should explain what happens to make his function correct, and what happens when it is incorrect. [2]

Accessing invalid memory location which could hold anything

When it works, it's because memory is holding a zero value, and adding that to the sum doesn't change anything [1 mk]

When it doesn't work, it's because that memory is holding some other non-zero value which does affect the sum [1 mk]

## Question 4 Programming [10 Marks]

1. Consider the function below:

```
vector<int> calculate(int N){
    int y = 2;
    vector<int> ans;
    while (y <= N){
        while (N % y == 0){
            ans.push_back(y);
            N = N / y;
        }
        y++;
     }
     return ans;
}
```

First, list the elements in the returned vector when the function is called with $N = 20$. Then, in plain words, explain the purpose of the function, and the meaning of the vector it returns. [3]

2, 2, 5 [1 mk]

Compute prime factors [2 mk] (1 mk for just saying factors)

2. Consider the function below that accepts a `vector` of integers and a single non-negative integer `N`, and returns the sum of the first `N` elements of the vector:

```
int sumTo(vector<int> numbers, int N){
    int sum = 0;
    for (int i = 0; i < N; i++){
        sum += numbers[i];
    }
    return sum;
}
```

Rewrite this function so that it is instead **recursive**. Your function may not contain any loops. [3]

```
int sumTo(vector<int> numbers, int N){
    if (N == 0){
        return 0;
    }
    return numbers[N-1] + sumTo(numbers, N-1);
}
```

1 for base case

1 for each term of the recursive call

-0.5 if it's off by one e.g. :

numbers[N] + sumTo(numbers, N-1);

3. Complete the C++ program below by filling in the blank lines. The program reads an integer value from the user and determines whether or not this integer is an Armstrong number. An Armstrong number is an integer that is equal to the sum of each of its digits to the power of the number of digits in the nymber. An example of a three-digit Armstrong number is 407, because $407 = 4^3 + 0^3 + 7^3$. [4]

```
1  #include <iostream>
2  #include <cmath>
3
4  using namespace std;
5
6  int main() {
7      int originalValue;
8      cin >> originalValue;
9      int currentValue = originalValue;
10     int sum = 0;
11     //calculate number of digits in the input:
12     int numDigits = log10(abs(originalValue)) + 1;
13
14     while (currentValue > 0){
15
16         int digit = currentValue % 10; //get the last digit
17
18         sum = sum + pow(digit, numDigits);
19
20         currentValue = currentValue / 10; //drop last digit
21     }
22
23     if (currentValue == originalValue){
24         cout << originalValue << " is Armstrong" << endl;
25     }
26     else {
27         cout << originalValue << " is not Armstrong" << endl;
28     }
29     return 0;
30 }
```

1 mk each

For line 16, could do complicated conversion to string -> get last char -> convert to digit
For line 20, could do some complicated conversion to string -> take substring except last char -> convert to int

If so, as long as correct, give mark

**This page intentionally left blank for extra space**