COS 214 Practical Assignment 1

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

- Date Issued: **26 July 2022**
- Date Due: **2 August 2022** at **11:00am**
- Submission Procedure: **Upload to ClickUP**
- Submission Format: **archive (zip or tar.gz)**

1 Introduction

1.1 Objectives

In this practical you will:

- revise C++ concepts and skills required for this module;
- revise recursion;
- revise pointer arithmetic;
- revise C++ templates; and
- implement the Memento pattern.

1.2 Outcomes

When you have completed this practical you should:

- understand and be able to use required C++ concepts for this module;
- understand and be able to implement functions recursively;
- apply the Memento to store the state of objects and re-instate the state at a later stage.

2 Constraints

1. You must complete this assignment individually.
2. You may ask the Teaching Assistants for help but they will not be allowed to give you the solutions.

3 Submission Instructions

You are required to upload all your source files (that is `.h` and `.cpp`), your Makefiles, and a single PDF document and any data files you may have created, in a single archive to ClickUP before the deadline.

4 Mark Allocation

Task	**Marks**
C++ Heap and Stack	15
C++ Function Dispatching	10
C++ Templates	20
C++ Pointer Arithmetic	15
Recursion	10
Design Patterns	35
TOTAL	105

5 Assignment Instructions

Task 1: C++ Heap and Stack ..... (15 marks)

Consider the following code listing:

```
int main()
{
    int a;
    double* b = new double;
    char c[10];
    long const n = 20;
    int d[n];
    const int* e = (const int*)522;
    void* f = (void*)0xacfe2675;
    char g = 2;
    const int h = NULL;
    c[10] = *&*e;
    cout << c[10]  << endl;
}
```

For all questions below, write up your answers in a PDF file for the assignment.

1.1 For each of the following variables — `a` - `h`, `n`, `c[10]` — state whether the variable is created on the stack or heap **AND** why. (10)

1.2 Why or why not would assigning the value of *NULL* to the variable `h` work? (2)

1.3 This code contains three coding errors that *could* result in the application not working as expected. **List** the three errors and state why they can/will/should result in an error in the application. (3)

Task 2: C++ Function Dispatching ..... (10 marks)

Given the following specification for classes:

- Create an abstract class called ClassA
- Create a class called ClassB
- Create derived ClassC from ClassA
- Create derived ClassD from ClassA and ClassB
- Create derived class E from ClassD
- Create a public empty constructor for each class that prints the following message to the console: `ClassX's Empty Constructor is Called`, where ClassX refers to the corresponding class
- Create a destructor for each class, which prints the message `ClassX's Destructor is Called`

For all questions below, write up your answers in a PDF file of the assignment.

2.1 When creating any derived class, when is the constructor of ClassA called? (2)

2.2 When destroying any derived class, when is the constructor of ClassA called? (2)

2.3 When creating the derived class ClassC, is the constructor called before or after the constructor of ClassA? (2)

2.4 When creating the derived class ClassD, in what order is the constructors called for ClassA and ClassB called? (2)

2.5 When destroying the derived class ClassD, in what order is the destructors called for ClassA and ClassB called? (2)

Task 3: C++ Templates ..... (20 marks)

Study the code listing used to build a calculator for integers:

```
class Calculator {
  private:
```

```
    int num1, num2;

  public:
    Calculator(int n1, int n2) {
        num1 = n1;
        num2 = n2;
    }

    int add() { return num1 + num2; }
    int subtract() { return num1 - num2; }
    int multiply() { return num1 * num2; }
    int divide() { return num1 / num2; }
};
```

3.1 Create two new files, called Calculator.h and Calculator.cpp. Generalise the above code to make use of any datatypes. Place your header definitions in the header.h file and your generalised calculator implementation into the Calculator.cpp. (8)

3.2 Using your new generalised calculator, if possible calculate and print the value of 741 / 13 to the console. Explain why this worked or did not. (3)

3.3 Using your new generalised calculator, if possible calculate and print the value of 127.58 + 54.971 to the console. Explain why this worked or did not. (3)

3.4 Using your new generalised calculator, if possible calculate and print the value of "Hello" + "World" + "!" to the console. Explain why this worked or did not. (3)

3.5 Using your new generalised calculator, if possible calculate and print the value of "Hello" * "World" + "!" to the console. Explain why this worked or did not. (3)

Task 4: C++ Pointer Arithmetic ..... (15 marks)

For the following questions, use the pointer notation **ONLY**. Do **NOT** use the array index [] notation.

4.1 Study the code below, predict **AND** explain the output each `cout` statement will give. (5)

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

typedef int *IntPtrType;

int main()
{
        IntPtrType ptr_a, ptr_b, *ptr_c;

        ptr_a = new int;
        *ptr_a = 15;
        ptr_b = ptr_a;
        cout << *ptr_a << " " << *ptr_b << "\n";

        ptr_b = new int;
        *ptr_b = 4;
        cout << *ptr_a << " " << *ptr_b << "\n";

        *ptr_b = *ptr_a;
        cout << *ptr_a << " " << *ptr_b << "\n";

        delete ptr_a;
        ptr_a = ptr_b;
        cout << *ptr_a << " " << *&*&*&*&*ptr_b << "\n";

        ptr_c = &ptr_a;
```

```
    cout << *ptr_c << " " << **ptr_c << "\n";

    delete ptr_a;
    ptr_a = NULL;

    return 0;
}
```

4.2 Implement the function `countEven(int*, int)` which receives an integer array and its size, and returns the number of even numbers in the array. Test your function using a main function. (5)

4.3 Implement the following function signature `double* maximum(double* a, int size);` that returns a pointer to the maximum value of an array of double's. If the array is empty, return NULL. Test your function using a main function. (5)

Task 5: Recursion ..... (10 marks)

5.1 Given the patterns below, implement a recursive function and main in a main.cpp for the input parameter $n > 0$: (5)

- Input = 20; Output = 20, 13, 6, -1, 6, 13, 20
- Input = 14; Output = 14, 7, 0, 7, 14

5.2 The Ackermann function was defined by Wilhelm Ackermann (1896-1962) in 1928. The Ackermann function is an example of a well-defined and total function, which is computable but not primitive recursive. *Well-defined and total* implies that the function itself doesn't break any of its piecewise definitions. Further it can be (theoretically) evaluated for any given set of inputs, making it *Computable*. A *Primitive recursive* function means that it can be computed using only for loops. The Ackermann function is however non-primitive recursive, which means that the number of required for loops isn't known. Rather the number of loops for a given iteration is part of the calculation itself. (5)

$$A(m,n) = \begin{cases} n+1 & \text{if } m = 0 \\ A(m-1,1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m-1, A(m,n-1)) & \text{if } m > 0 \text{ and } n > 0 \end{cases}$$

Implement the Ackermann function and the main in a main.cpp file, **calculate and display on the console** the value of A(4,2).

Task 6: Design Patterns ..... (35 marks)

A requirement on a login system is to allow users to set and update their passwords at will on their account. You need to find a secure way to allow the user to rollback to old passwords. Having learned about the benefits of the Memento pattern in your Computer Science studies, you decided to proceed with this pattern. You have been provided with a partial implementation attempt from a work colleague of the Memento pattern which was adapted from an example provided on `https://refactoring.guru/design-patterns/memento`. Your work colleague proceeded to split the code into various classes but was unable to complete the requirement. Study the code and answer the questions that follow.

6.1 Which class/interface in the provided files is equivalent to the Memento interface in the Memento pattern? (2)

6.2 Which class/interface in the provided files is equivalent to the ConcreteMemento class in the Memento pattern? (2)

6.3 Which class/interface in the provided files is equivalent to the Originator class in the Memento pattern? (2)

6.4 Which class/interface in the provided files is equivalent to the Caretaker class in the Memento pattern? (2)

6.5 For each of the provided files, create the corresponding header files and setup appropriate using clauses. (5)

6.6 Complete the implementation of the Memento pattern for the given files to satisfy the above requirement. (10)

6.7 Use Visual Paradigm to model the given code as a class diagram. Add the exported class diagram to your submitted PDF file. (12)