UOW AUSTRALIA

School of Computing and Information Technology

Family name	
Other names	
Student number	
Table number	

CSCI251
Advanced Programming

Examination Paper
Autumn Session 2021

Exam duration	**3** hours (except for some RA students)
Weighting	60%
Directions to students	This exam contains two parts, for a total of 40 marks. Part A: 16 questions worth 1 mark each. Part B: 12 questions worth 2 marks each.

All code provided in this exam is assumed to have appropriate headers. In writing code you should not provide appropriate includes, they will be assumed, unless the question specifically relates to includes.

Answer your questions in a file and convert the file to pdf.

Name your pdf file with your name and student number, as in **Ngoc_007.pdf**.
DO include your name and student number of each page of your answer paper.
DO NOT include the question itself, just the part and number clearly identifying which question an answer relates to.

Part A: 16 questions worth 1 mark each. (Total 16 Marks)

1) What do exceptions allow us to separate? Give an example of when this separation is appropriate.

2) Explain what the idea that **`main()`** should tell a story means, particularly in procedural programming.

3) Function **`swapTwo()`** below is used by which method: pass by value or pass by reference? Explain what is the main difference between two methods.

```
void swapTwo(int& a, int& b){
       int tmp = a;
       a = b;
       b = tmp;
}
```

4) How many types of overloading in C++? Give examples to illustrate.

5) The following code segment fails to initialize the array elements to 1. State the usage of the keyword auto. Rewrite the code segment so that it could work.

```
int *ptr = new int[7];
for(auto& x : ptr)
       x = 1;
```

6) Show two ways to insert **"comments"** in C++ code

7) An array name is often considered as a *"constant pointer"* to an array. Consider the following statements and state whether there is any difference in terms of the byte size of **`ptr1`** and **`ptr2`**.

```
int ptr1[5];
int * const ptr2 = new int[5];
```

8) What a loop in C++ is used for and when you would use one?

9) What is the role of a makefile? How to produce a makefile?

10) What are the roles of constructors and destructors, related to the lifetime of an object? Give an example header for a constructor and destructor for a class X.

11) How do static and dynamic libraries differ? Give one advantage of using each.

12) What does it mean to provide **`.h`** and **`.o`** files to a “client”? Why would we do this?

13) State one advantage of smart pointers (not iterators from STL) over regular pointers.

14) Explain the idea of class templates being blueprints of blueprints.

15) Describe a scenario where the use of an STL **`vector`** makes more sense than the use of an STL **`deque`**, STL **`set`** or STL **`map`**. Justify your answer.

16) Describe advantages of using a container class rather the classical array.

Part B: 12 questions worth 2 marks each. (Total 24 Marks)

1) Assume that Test is a class provided so the code below compiles, and that Test doesn't generate additional internal **`Test`** objects. Explain where and how many **`Test`** objects will be created by the following code. If you provide the correct number with no justification you will receive **`0.5`** mark.

```
void testOne(Test &C)
{
	cout << "Testing" << endl;
}
void testTwo(Test C)
{
	cout << "More testing" << endl;
}

int main()
{
	Test A;
	Test B = A;
	testOne(A);
	testTwo(B);
	return 0;
}
```

2) Write the definition and implementation of a class **`Frog`**, which is derived from **`Animal`**. The base class contains private data fields for the name and age, a public constructor that sets both of those data fields, and getter functions for each of the data fields. The derived class contains private fields for the tongue length, mass, and the hop distance. The derived class also contains a constructor for setting all values, and an overloaded insertion operator. You do not need to write the base class.

3) You will write 3 small blocks of code in the sections (**TODO…**) to demonstrate move constructor. You do not need to write the whole program. The code and the running result below are the suggestions for your code.

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

class Mouse
```

```
{
private:
   string name;
   int age;

public:
   Mouse() = default;
   ~Mouse();
   Mouse(string name, int age);
   Mouse(const Mouse&); //copy constructor
   TODO...; //move constructor
};

Mouse::~Mouse()
{
  cout << "Destructor" << endl;
}

Mouse::Mouse(string _name, int _age)
{
   name = _name;
   age = _age;
   cout<<"Mouse constructing"<<endl;
}

Mouse::Mouse(const Mouse& copyMouse)
{
   name = copyMouse.name;
   age = copyMouse.age;
   cout<<"Copy Mouse constructor"<<endl;
}

Mouse::Mouse(Mouse && mMouse){
    TODO....
    cout<<"Move Mouse constructor"<<endl;
}

int main()
{
    Mouse myMouse("Katty", 2);
    Mouse secondMouse=myMouse;
    TODO........;// call move constructor
    return 0;
}
```

```
Mouse constructing
Copy Mouse constructor
Move Mouse constructor
Destructor
Destructor
Destructor
```

4) Consider that **`Cat`** and **`Animal`** are related in an inheritance hierarchy. Write code to demonstrate an appropriate use of protected data.

5) Explain the purpose of **`const`** and **`static`** qualifiers for data members and for member functions. This means you should be covering four points.

6) Declare a Circle class, which has the data field of radius and the member function of area to calculate the area enclosed by this circle. Write the implementation to create a circle with radius = 1.2 and display its area.

7) Consider that we have three classes, **`Cat`**, **`House`**, and **`HouseCat`**. Describe two possible relationships between the three classes. For each relationship include a sketch of code as part of your explanation. Explain why each of the two relationships is or isn't a reasonable one.

8) Explain how aggregation and composition differ. Give examples to support your explanation.

9) Write code to produce function objects to take and add two floats and a double, and return the result as an int. illustrate how you would make and use this function object.

10) Write some lines of code in the section (**TODO…**) to create a function template to **find the minimum value of an array**. You do not need to write the whole program. The main function and the running result below are the suggestions for your code.

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

template<typename T>
T findMinArr(T* arr, int s){
   TODO...
}

int main(){
    int a[]={7, 8, 1, 2, 4, 12, 4};
    char b[]={'h', 'i', 'a','o','w'};
    double c[]={7.77, 1.11, 2.22, 3.33, 8.88};

    cout<<"Min value of INT array: "
        << findMinArr(a, sizeof(a)/sizeof(int))<<endl;
    cout<<"Min value of CHAR array: "
        << findMinArr(b, sizeof(b)/sizeof(char)) <<endl;
    cout<<"Min value of DOUBLE array: "
        << findMinArr(c, sizeof(c)/sizeof(double)) << endl;
    return 0;
}
```

```
Min value of INT array: 1
Min value of CHAR array: a
Min value of DOUBLE array: 1.11
```

11) Write a function template **`Worst`** to return the worst of 3 objects of the same type, for a sensible meaning of worst that you need to specify. State what types would need so your function template can be applied to them. Write a class **`Bike`**, with data fields you choose, with functionality such that the function template can be applied to three **`Bike`** objects. Write driver code illustrating the use of **`Worst`** on three **`Bike`** objects.

12) Explain the relationship between containers and iterators. Give examples for one sequential and one associative container separately and explain briefly how we might appropriately use each.

END of EXAMINATION