COMP 2404 ABC Winter 25
Practice Final

These are the instructions you will see on exam day.

- Please make sure your **name** and **student number** are on the **bubble sheet**.
- There are **44 multiple choice** questions which you should answer on the **bubble sheet**.
- The first **28 questions** are general knowledge questions.
- The next **16 questions** are application programming questions. You will be given a set of classes that work together as an application. You will answer questions on how to implement these functions correctly, using the classes and functions given, and in a way that correctly manages memory.

Marking Scheme: This exam is out of **60 marks**.

- [28 marks] The **28 general knowledge questions** are worth **1 mark each**.
- [32 marks] The **16 application programming questions** are worth **2 marks each**.
- You have 2 minutes per mark, plan your time accordingly.

1 Multiple Choice

Given the following piece of code, select all lines which could go in the locations marked “//code goes here” that would NOT cause a compiler error, or select “None of these” if they would all cause a compiler error.

wheels++;

doors++;

pistons++;

None of these.

```
class Car{
  public:
    int wheels;
  protected:
    int doors;
  private:
    int pistons;
};

class Electric: public Car{};

class Tesla: public Electric{
  void upgrade(){
    //code goes here
  }
 };

int main(){
  Tesla ts;
  //code goes here
  return 0;
}
```

Consider putting different access modifiers (i.e., private inheritance). What lines would not cause a compiler error now?

What would be output by the following piece of code (assume proper #includes statements, etc. are there)?

```
class Animal { };
class Bird : public Animal { };
class Chicken : public Bird { };
class Cat : public Animal { };
class Pig : public Animal { };

int main(){
  try {
      throw Pig();
  } catch(Bird& a) {
    cout<<"Bird alert!"<<endl;
  } catch(Chicken& a) {
    cout<<"Chicken alert!"<<endl;
  } catch(Animal& a) {
    cout<<"Animal alert!"<<endl;
  } catch(Pig& a) {
    cout<<"Pig alert!"<<endl;
  } catch(...){
    cout<<"Caught something"<<endl;
  }
    return 0;
}
```

What would be the output if we changed the order of the catch blocks? What would be the output if we threw a different object, or an int or a string?

Write a copy constructor for the Show class that does a deep copy.

How would the copy constructor change for different types of arrays?

```cpp
class Episode{
    public:
        Episode(const string& name, const string& captions);
    private:
        string name;
        string captions;
};

class Show {
    public:
        Show(const string& name): name(name), numEpisodes(0), episodes(new
Episode*[256]){}
        Show(Show& show);
        void addEpisode(const string& name, const string& captions){
            episodes[numEpisodes++] = new Episode(name, captions);
        }
    private:
        string name;
        Episode** episodes;
        int numEpisodes;
};
```

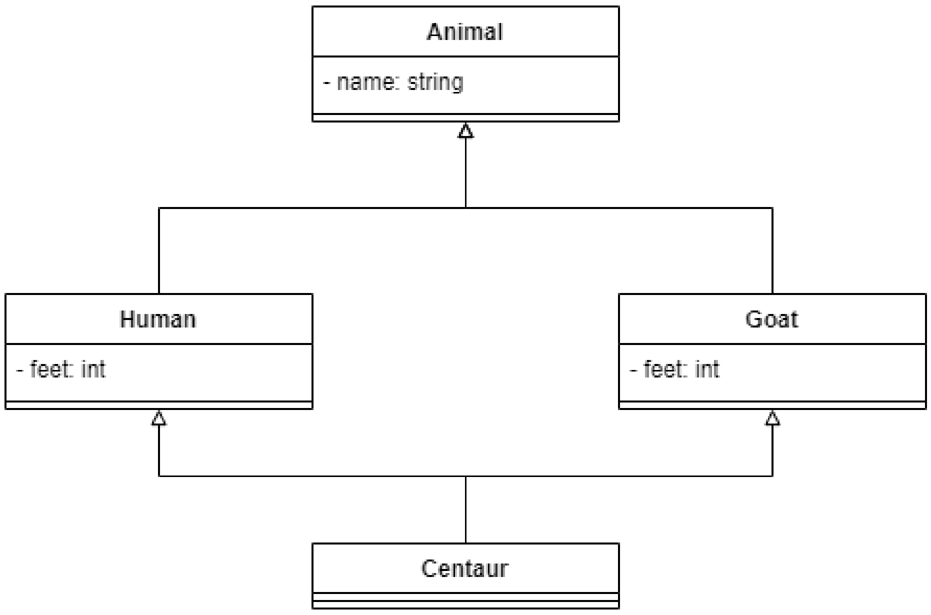
```cpp
using namespace std;

class Animal{
    public:
        Animal(){cout<<"Animal ctor"<<endl;}
        ~Animal(){cout<<"Animal dtor"<<endl;}
};
class Bird: public Animal{
    public:
        Bird(){cout<<"Bird ctor"<<endl;}
        ~Bird(){cout<<"Bird dtor"<<endl;}
};

int main(){
    Animal animal;
    Bird bird;
    return 0;
}
```

What would be the output of this code? What if there were more animals in the class hierarchy? What if the Bird class contained an Egg object that also printed out when it was in the constructor and destructor?

```cpp
class Date {
    public:
        Date(int = 1901, int = 1, int = 1);
        Date(Date&);

        //setters
        void setDay(int);
        void setMonth(int);
        void setYear(int);
        void setDate(int, int, int);
        void setDate(Date&);

    private:

        int day;
        int month;
        int year;

};
```

How would you change the function signature and write the setters to implement *cascading* in the Date class above?

Other questions:



Recall the diamond hierarchy from multiple inheritance. An example is given above. What do the constructors for each class look like? Assume Animal does NOT have a default constructor. What do the constructors look like now? What if none of these have default constructors? What member variables does each class have? What difference does using virtual inheritance make?

```
class Time{
    public:
      Time(int = 0, int = 0, int = 0);

    private:
      int seconds, minutes, hours;
      int convertToSeconds();
 };
```

How would you implement the + operator for this class? What if we wanted to implement the == operator? What if we also wanted to compare the Time class to an int using the == operator?

2 Additional Multiple Choice Hints

1. What things are represented in a UML diagram? What things are NOT represented in a UML diagram?
2. What are the object design categories, and what are some examples?
3. What are some design patterns and where might they be used?
4. You should know basic iterator use, such as a simple for-loop, and how you would access the data using an iterator.
5. You should know **very well** what is the different between a variable, a pointer, and a reference, and how they might be used together. In particular, if I declared pointers, references, and variables in various ways, you should know how many pieces of data actually exist, and the different ways you could access that data.

3 Programming

There are four main parts to the programming section. All use the same entity classes, given below. Make sure you can write the function for the stream insertion operator (declared as a friend function) so that it will play the movies using polymorphism.

`Criteria` is an abstract class. You may assume that appropriate concrete classes exist.

```
#define MAX_ARR 200
#include <iostream>
#include <string>

using namespace std;

class Movie{
    friend ostream& operator<<(ostream&, const Movie&);
    public:
        Movie(const string& title, float price);
        bool operator==(const string& t) const {return title = t;}
    protected:
        virtual void play(ostream& ost) const = 0;
        string title;
        float price;
};

// high definition
class HDMovie: public Movie{
    public:
        HDMovie (const string& title, float price, const string& hd);
    protected:
        virtual void play(ostream& ost);
        string hdContent;
};

// standard definition
class SDMovie: public Movie{
    public:
        SDMovie (const string& title, float price, const string& sd);
    protected:
        virtual void play(ostream& ost);
        string sdContent;
};


// returns true if the Movie matches the search criteria
class Criteria {
    public:
        virtual bool matches(const Movie&) const = 0;
};
```

3.1 Programming Linked Lists

```
class MovieList{

    class Node{
        public:
            Movie* data;
            Node* next;
    };

    public:
        MovieList();
        ~MovieList();

        // add to the MovieList at the index indicated
        void add(Movie*, int index);

        // remove and return the Movie* from the index indicated
        Movie* remove(int index);
    private:
        Node* head;
};
```

This class contains a sequence of movies the user would like to watch. This is a singly linked list. Make sure you can write each of these functions. The `MovieList` should not delete data.

3.2 Programming with Templates

```
// Array.h
#ifndef ARRAY_H
#define ARRAY_H
#include <iostream>
#include <string>
#define MAX_ARRAY 256

using namespace std;

template <class T>
class Array {
    public:
        Array();
        ~Array();
        // add T to the back of the array if there is room
        Array<T>& operator+=(const T&);
        // remove T if it exists
        Array<T>& operator-=(const T&);
        // return element at index, or call exit(1) on bad index
        T& operator[](int index);
        const T& operator[](int index) const;
        // inline functions
        int size() const { return numElements; }
        bool isFull() const { return numElements >= MAX_ARRAY;}

    private:
        int numElements;
        T* elements;
};
// your code would go here
#endif
```

Be sure you can write each of these functions including the constructor and destructor.

```
class MovieFlix{
    public:
        MovieFlix(const string& subscriber);

        ~MovieFlix();

        // make a new Movie and add it to the array
        bool addMovie(const string& title, float price,
            const string& content);

        // get the Movie with the title given
        Movie* getMovie(const string& title) const;

        // remove the Movie with the title given
        Movie* removeMovie(const string& title) const;

        // add all movies that match the
        // Criteria object to the MovieList
        void addToPlayList(const Criteria&);

        // remove each Movie from (the front of) the MovieList
        // and "play" (print) it to the ostream
        void playMovieList(ostream& ost);

    private:
        // the name of the MovieFlix subscriber
        string subscriber;

        // Collection class which stores the Movies
        Array<Movie*> movies;

        // The current playlist of movies
        MovieList playlist;
};
```

You should be able to write any of these functions. You should take into consideration proper memory management in both this class, the `Array` class, and the `MovieList` class. Code that would result in memory leaks or double frees will lose marks.

Logic and understanding are more important than syntax. You will NOT be penalized for **minor** syntax mistakes like missing semi-colons or brackets. However, you may lose minor marks for incorrectly using pointers and references.