CMPS 373 Exercises

Question 1 Consider the following template definition in C++:

```
template < class T > class cell
{
      protected:
            T info;
      public:
            void set(T x){ info = x; }
            T get() { return info; }
};
```

Define the subclass `colored_cell` by extending the class cell with:

- a field color, indicating the color of the cell, represented as a character ("w" for white, "b" for black, "r" for red, etc.),
- the method `set_color`, which set the content of the field color,
- the method `get_color`, which returns the content of the field color, and
- an updated method `get`, which returns the content of the field info if the color is not white, and returns 0 otherwise.

Choosing the right signature, types and parameters is part of the exercise.

Question 2 We want to implement in C++ stacks and queues as linked lists, using templates and inheritance.

1. Define a class element with the following fields:
    a. a field info of parametric type T (the information contained in the element)
    b. a field next of type pointer to element (the next element in the list)
2. Define a template class `stack_or_queue` parametric on T, containing element as an *internally* declared class, and
    a. a method `pop` which removes the first element of the list and returns its information
    b. a method `is_empty` which returns true or false depending on whether the list is empty or not
3. Define a template class `stack` (parametric on T) as subclass of `stack_or_queue`, with an additional method `push` which adds a new element as first element of the list.
4. Define a template class `queue` (parametric on T) as subclass of `stack_or_queue`, with an additional method `insert` which adds a new element as last element of the list.

Defining appropriate constructor and destructor methods is part of the exercise. All the methods, except the destructors, should be executed in constant time (*i.e.* no scanning of the list should be required). You can of course use additional (private) fields and methods if you find them useful. Take care of protecting the implementation correctly.

Question 3 In C++, write the member function to overload the function call operator. Give the parameter the value of the data member `cageNumber`. Basically the same functionality as a getter method.

```
class ZooAnimal
{
	private:
		char* name;
		int cageNumber;
		int weightDate;
		int weight;
	public:
		...
		void operator() (int&);
		...
};

int main ()
{
	ZooAnimal bozo ("Bozo", 408, 1027, 400);
	int cage;
	bozo(cage);
	cout << cage << endl;
}
```

Question 4 Write the function that defines the overloaded >> operator for ZooAnimal objects to input values for the name, cage number, weight date, and weight (respectively) of the ZooAnimal object parameter.

```
class ZooAnimal
{
	private:
		char *name;
		int cageNumber;
		int weightDate;
		int weight;
	public:
		ZooAnimal (char*, int, int, int);
		inline ~ZooAnimal () { delete [] name; };
		void changeWeight (int pounds);
		char* getName ();
		int getWeight ();
		int daysSinceLastWeighed (int today);
		friend ostream& operator<< (ostream& stream, ZooAnimal& Z);
		friend istream& operator>> (istream& stream, ZooAnimal& Z);
};
```

Question 5 From an object oriented perspective is it still George Washington's ax if the handle has been replaced 5 times and the head 4?

Question 6 How does inheritance break the encapsulation barrier around a class and increase coupling?

Question 7 What does it mean to program to an interface rather than an implementation? Why is this desirable?

Question 8 Is private class data also secure data?

Question 9 In C++, list three ways we can gain access to private members while not being in the scope of the class.

Question 10 Consider the following C++ program:

```
class X
{
      public:
            X(int p) : fx(p) {}
            int fx;
};
class Y
{
      public:
            Y(int p) : fy(p) {}
            int fy;
};
class B : public virtual X,public Y
{
      public:
            B(int p) : X(p-1),Y(p-2){}
};
class C : public virtual X,public Y
{
      public:
            C(int p) : X(p+1),Y(p+1){}
};
class D : public B, public C
{
      public:
            D(int p) : X(p-1), B(p-2), C(p+1){}
};
int main()
{
      D* d = new D(5);
      B* b = d;
      C* c = d;
      std::cout << b->fx << b->fy << c->fx << c->fy;
      return 0;
}
```

What is the output of running the program?

Question 11 What is the following C++ code doing? In particular, the `marry` method.

```
class Person
{
      Person *spouse;
      string name;

      public:
            Person (string n) { name = n; spouse = nullptr; }
            bool marry (Person *p)
            {
                  if (p == this) return false;
                  spouse = p;
                  if (p) p->spouse = this;
                  return true;
            }
            Person *getSpouse () { return spouse; }
            string getName () { return name; }
};
```

Question 12 Consider the following C++ code:

```
class A {
      public:
            A(int i) { std::cout << "A" << i; }
            A() { std::cout << "A1"; }
            virtual int get() { ... }
};
class B: public A {
      public:
            B(int i) : A(i) { std::cout << "B" << i; }
};
class C: public A {
      public: C(int i) : A(i) { std::cout << "C" << i; }
};
class D: public B, public C {
      public:
            D(int i) : B(i + 10), C(i + 20) { std::cout << "D" << i; }
};
```

Why does the following client code not compile?

```
void client()
{
      D* d = new D(5);
      std::cout << d->get();
}
```