CMPSC 32 W19
Object Oriented Design and Implementation

Final Examination

Please state your answers as clearly as possible. This exam not only tests your understanding of the material, but also how well you can convey your understanding to us. Remember that you are solely responsible for the answers to the questions, therefore, please refrain from consulting with your class peers.

Please write all your answers **LEGIBLY** and **CLEARLY**. If we cannot decipher your answers, you will not receive credit.

No electronic devices are allowed during the exam (calculators, cell phones, laptops, etc.).

READ all questions carefully before attempting to answer. If there are any ambiguities in the statement of questions, please ask us. **You may assume that each problem is correct and solvable unless the question specifically asks about errors.**

THE GRADE IN THIS EXAM IS A TOTAL OF 80 POINTS.

Name (as it would appear on the official course roster)	Umail Address
	@umail.ucsb.edu

Question 1 (6 points) Write whether each statement is True or False. If False, briefly state why.

a. In a multi-core processor, a single-threaded process can only execute on a single core at any given time.

b. The best-case running time for the quicksort algorithm discussed in class is O(n).

c. A subclass can access a base class' private members directly by name (without an "accessor" method).

d. C++ structs support inheritance and polymorphism.

e. Anything that is not declared in a programmer-defined namespace is automatically put into the global namespace.

f. In a `std::vector`, accessing an index that is out-of-bounds using the `[]` operator will throw an exception.

Question 2 (11 points)

a. **Briefly** describe the difference between the `ps` and `jobs` Unix commands. Assuming we have a Unix executable named `myProgram` in the current directory, write the Unix command that will make the `myProgram` process appear when executing both the `jobs` and `ps` commands (assuming `myProgram` is currently running).

b. **Briefly** describe what a race condition is the context when two threads share a resource (a value in memory).

c. **Briefly** define a priority queue. Why is it more efficient to use a heap as a priority queue than to use a sorted list as a priority queue?

d. **Briefly** define what a normal, boundary, and error case is when writing unit tests as discussed in lecture.

Question 3 (12 points)

Write the output for the following application. **For each line of output, write a comment next to each line stating which process or thread outputs the line (read below for thread notation)**. You may assume that:

- Processes are defined in the order the OS created them. For example, **P1** is the process the OS created when executing the application, **P2** is the second process the OS created, etc. When forking a process, you may assume the parent process has priority and executes a statement before a child process.

- Each process has a main thread, and any thread may create more threads. Threads are defined in the order the OS created them and are associated with a specific process. For example, if P1's main thread created a child / subthread, then this is denoted as **[P1, T1]**. If P1 (or a thread in P1) then creates another thread, then this thread is denoted as **[P1, T2]**. Any output that the main thread produces is simply denoted with just the process (for example: **[P1]**). Use this notation in your comments when stating which thread outputs a line to the console.

- The OS executes one instruction from each active main thread or subthread in the order that the processes were created. For example, if there are three active processes and two active subthreads in P1, then the OS executes one instruction from P1, then one instruction from [P1, T1], then one instruction from [P1, T2], then one instruction from P2, then one instruction from P3, and then one instruction from P1, etc.

- The time for the OS to create and start executing a thread is less than 1ms.

- Any instruction (except sleep) is executed in less than 1 ms.

```
#include <iostream>
#include <unistd.h>
#include <thread>

using namespace std;

void f() {
     sleep(5);
     cout << "f()" << endl;
}

int main() {
     cout << "starting main" << endl;
     pid_t x = fork();
     cout << "after fork1" << endl;
     if (x != 0) {
          sleep(2);
          cout << "x != 0" << endl;
          thread a(f);
          a.join();
          cout << "thread a done" << endl;
     } else {
          cout << "x == 0" << endl;
          pid_t y = fork();
          cout << "after fork2" << endl;
     }
     sleep(1);
     cout << "end" << endl;
}
```

Question 4 (14 points)

A partial class definition for an `IntegerList` is given below. This `IntegerList` implementation manages an array of ints on the heap. The size of the array is defined by `CAPACITY` and will throw an exception if the number of elements exceeds the capacity. Assuming the maintenance of `IntegerList` is implemented correctly, complete the methods for the “Big Three” (destructor, copy constructor, and assignment operator). Note that you must implement a deep copy of the elements and handle self-assignment correctly. Your solution must be syntactically correct in order to receive full credit.

```
#ifndef SIMPLELIST_H
#define SIMPLELIST_H

class FullListException {};

class IntegerList {
public:
      static const int CAPACITY = 10;
      IntegerList() { numElements = 0; elements = new int[CAPACITY]; }
      ~IntegerList();                                  // implement destructor
      IntegerList(const IntegerList &orig);            // implement copy constructor
      IntegerList& operator=(const IntegerList &right); // implement assignment op.
      int* getList() const { return elements; }
      int getNumElements() const { return numElements; }
      void insert(int item) throw (FullListException);
private:
      int numElements;  // number of elements in the IntegerList
      int* elements;          // Dynamic array containing ints
};

#endif
// IntegerList.cpp (write the Big Three methods below in the .cpp file)
```

Question 5 (14 points)

a. Write the resulting array for a MaxHeap implementation discussed in class after the following code segment. Note, you may assume that the MaxHeap array has "junk" data in index 0.

```
MaxHeap mh;
mh.insert(10);
mh.insert(5);
mh.insert(15);
mh.insert(20);
mh.insert(8);
mh.removeMax();
mh.removeMax();
mh.insert(12);
mh.insert(30);
```

b. Complete the algorithm for the `MaxHeap::removeMax` and `MaxHeap::heapify` methods in the lecture notes by filling in the blanks with the proper expression, operator, or value.

```
int MaxHeap::removeMax() throw (HeapEmptyException) {
      if (size _______________________ 0)
            throw HeapEmptyException();
      if (size _______________________ 1) {
            size--;
            return heapArray[_______________________];
      }
      int index = 1;
      int max = heapArray[index];
      heapArray[_______________________] = heapArray[_______________________];
      size--;
      heapify(_______________________);
      return _______________________;
}
void MaxHeap::heapify(int index) {
      int leftChild = _______________________;
      int rightChild = _______________________;
      int largestIndex = _______________________;
      if (leftChild <= size &&
          heapArray[leftChild] > heapArray[_______________________]) {
            largestIndex = _______________________;
      }
      if (rightChild <= size &&
          heapArray[rightChild] > heapArray[_______________________]) {
            largestIndex = _______________________;
      }
      if (largestIndex != _______________________) {
            int temp = heapArray[_______________________];
            heapArray[index] = heapArray[_______________________];
            heapArray[largestIndex] = temp;
            heapify(_______________________);
      }
}
```

Question 6 (8 points)

Using the following class definitions and various definitions for function `f()`, write the output for parts **a – d**. You may assume that each part executes independently of the other parts. If the segment of code results in a compilation error, briefly state why. If the segment of code has a runtime error, write the output up to the point of the error and briefly state why the runtime error occurred.

```
#include <iostream>
using namespace std;
class A {
public:
            A() {}
            virtual void f2() { cout << "A.f2" << endl; }
};
class B : public A {
public:
            B() {}
            virtual ~B() { cout << "~B" << endl; }
            virtual void f2() { cout << "B.f2" << endl; }
            virtual void f3() = 0;
};
class C : public B {
public:
            C() {}
            ~C() { cout << "~C" << endl; }
            virtual void f2() { cout << "C.f2" << endl; }
            void f3() { cout << "C.f3" << endl; }
};
class D : public B {
public:
            D() {}
            ~D() { cout << "~D" << endl; }
            virtual void f1() { cout << "D.f1" << endl; }
            void f3() { cout << "D.f3" << endl; }
};
void g(B* b) { b->f2(); b->f3(); }
```

a.

```
void f() {
      A* a = new C();
      a->f3();
}
```

b.

```
void f() {
      C* c1 = new C();
      D* d1 = new D();
      g(c1);
      g(d1);
}
```

c.

```
void f() {
      D d;
      g(&d);
}
```

d.

```
void f() {
      B b;
      C c;
      b = c;
      g(&d);
}
```

Question 7 (10 points)
For the following code, write the resulting output.

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

class A {};
class B {};
class C : public B {};
class D : public A {};

void X(int value) throw (A,B,C,D) {
      if (value == 1) { throw D(); }
      else if (value == 2) { throw B(); }
      else if (value == 3) { throw C(); }
      else if (value == 4) { throw A(); }
}
void Y(int value) {
      try { X(value); }
      catch (C e) { cout << "Caught Y:C" << endl; }
      catch (D e) { cout << "Caught Y:D" << endl; }
}
void Z(int value) {
      try { Y(value); }
      catch (B e) { cout << "Caught Z:B" << endl; }
      catch (A e) { cout << "Caught Z:A" << endl; }
}


int main() {
      try {
            cout << "starting main" << endl;
            int array[] = {1,2,3,4};
            for (size_t i = 0; i < 4; i++) {
                  cout << array[i] << endl;
                  Z(array[i]);
            }
      }
      catch (A e) { cout << "main:A" << endl; }
      cout << "end of main" << endl;
}
```

Question 8 (5 points)
Given the following struct definition:

```
struct X { char a; char b; int c; char d; short e; };
```

Write the output for each `cout` statement in the blank spaces for the following code segment:

```
X a;
cout << &a << endl;                               0xc477b8
cout << reinterpret_cast<void*>(&a.a) << endl;    ____________
cout << reinterpret_cast<void*>(&a.b) << endl;    ____________
cout << &a.c << endl;                             ____________
cout << reinterpret_cast<void*>(&a.d) << endl;    ____________
cout << &a.e << endl;                             ____________
```

Scratch Page