CMPSC 32 F18
Object Oriented Design and Implementation

Midterm 2 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 41 POINTS.

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

Question 1 True / False (7 points)

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

a. Mergesort's worst-case running time is $O(n^2)$.

b. Within a try / catch mechanism, multiple catch blocks can be executed when an exception is thrown.

c. Class constructors cannot be inherited.

d. A base class' constructor is automatically called when constructing a subclass object.

e. Quicksort's worst-case running time is $O(n^2)$.

f. An abstract class must define all of its member functions as pure virtual functions.

g. Only objects that are inherited from the Exception class can be thrown.

Question 2 (6 points)
For the quicksort implementation covered in class, complete the algorithm by filling in the blanks with the proper expression or values.

```
void partition(int a[], size_t size, size_t& pivotIndex) {
      int pivot = a[0];        // choose 1st value for pivot
      size_t left = 1;         // index just right of pivot
      size_t right = size - 1; // last item in array
      int temp;

      while (left <= right) {

            while (_____________________________________________________) {

                  left++;
            }

            while (_____________________________________________________) {

                  right--;
            }
            if (left < right) {
                  temp = a[left];
                  a[left] = a[right];
                  a[right] = temp;
            }
      }

      pivotIndex = _____________________________;

      temp = a[0];
      a[0] = a[pivotIndex];
      a[pivotIndex] = temp;
}

void quicksort(int a[], size_t size) {
      size_t pivotIndex;      // index of pivot
      size_t leftSize;        // num elements left of pivot
      size_t rightSize;       // num elements right of pivot

      if (size > 1) {

            partition(_______________,_______________,_______________);

            leftSize = ____________________________________________;

            rightSize = ____________________________________________;

            quicksort(_____________________________,_____________________________);

            quicksort(_____________________________,_____________________________);
      }
}
```

Question 3 (11 points)

Three class definitions and a main function are given below:

```
class A {
public:
	~A() { std::cout << "~A()" << std::endl; }
	void f1() { std::cout << "A.f1()" << std::endl; }
	void f2() { std::cout << "A.f2()" << std::endl; }
	virtual void f3() = 0;
};

class B : public A {
public:
	B() {}
	virtual void f1() { std::cout << "B.f1()" << std::endl; }
	void f2() { std::cout << "B.f2()" << std::endl; }
	void f3() { std::cout << "B.f3()" << std::endl; }
};

class C : public B {
public:
	C() { c = new int(0); }
	~C() { delete c;  std::cout << "~C()" << std::endl; }
	void f1() { std::cout << "C.f1()" << std::endl; }
	void f2() { std::cout << "C.f2()" << std::endl; }
	void f3() { std::cout << "C.f3()" << std::endl; }
	int* c;
};

int main() {
	f();
	return 0;
}
```

The parts below provide a definition for the function `f()` called in `main`. For each part, **1)** write the output of the program using the provided definition for `f()`, **2)** if the code will result in a compilation error, write **ERROR** and *briefly* state why, and **3)** if a memory leak exists, *briefly* state where the leak exists.

a.

```
void f() {
	B b;
	A a = b;
	a.f2();
}
```

b.

```
void f() {
	A* a = new C();
	a->f1();
	a->f2();
	a->f3();
	delete a;
}
```

c.

```
void f() {
	C c;
	B b = c;
	b.f1();
	b.f2();
	b.f3();
}
```

d.

```
void f() {
	B* b = new C();
	A* a = b;
	a->f1();
	a->f2();
	a->f3();
	delete a;
}
```

Question 4 (4 points)

Three class definitions and a main function are given below:

```
class X {
public:
       X() { a = 10; b = 20; }
       short a;
       int b;
};


class Y : public X {
public:
       Y() : X() { c = 30; d = 40; }
       short c;
       short d;
};



int main() {
       X x;
       Y y;
       X z = y;
       return 0;
}
```

Three objects are created in `main`. Draw each object's memory structure (`x,y,z`) in the space above assuming that object `x`'s memory address is `0x7fff5d8f9528`, `y`'s memory address is `0x7fff5d8f9518` and `z`'s memory address is `0x7fff5d8f9510`. For each class' member variables, write each of their memory address location.

Question 5 (7 points)

a. **Briefly** (~ 1 – 2 sentences each) define *public*, *protected*, and *private* inheritance.

b. **Briefly** (~1 – 2 sentences each) define *normal*, *boundary*, and *error* case tests.

c. **Briefly** define what a *callback* is.

Question 6 (6 points)
Given the code below:

```
class ExceptionA {};
class ExceptionB : public ExceptionA {};
class ExceptionC : public ExceptionA {};

void f1(int x) throw (ExceptionA, ExceptionB) {
	if (x % 2 != 0)
		throw ExceptionA();
	else
		throw ExceptionB();
}

void f2(int x) throw (ExceptionC, ExceptionA) {
	if (x <= 0)
		throw ExceptionC();
	else
		throw ExceptionA();
}

void f3(int x) throw (ExceptionB, ExceptionC) {
	if (x < 10)
		throw ExceptionB();
	else
		throw ExceptionC();
}

void f(void (*funcPointer) (int), int value) {
	try {
		funcPointer(value);
	} catch (ExceptionB) {
		cout << "Caught ExceptionB" << endl;
	} catch (ExceptionA) {
		cout << "Caught ExceptionA" << endl;
	} catch (ExceptionC) {
		cout << "Caught ExceptionC" << endl;
	}
}
```

Write the output using the following main function:

```
int main() {
	f(f3, -10);
	f(f3, 10);
	f(f2, -10);
	f(f2, 10);
	f(f1, 20);
	f(f1, 11);
	return 0;
}
```

Scratch Paper (Do not detach)