ITE1015 Creative Software Design

Fall 2021 Midterm Exam

ID	NAME

Notice

1. Please check if you received all 8 pages of the test material excluding the title page.
2. Write down your student ID and name on the test material and answer sheets.
3. Write down page numbers on your answer sheets.
4. Write down your answers using a black or blue ballpoint pen
5. You can use any C++ grammar and STL unless they are not prohibited in the problem description.
6. You must include all the necessary headers to get full credits.
7. You can leave from 1 hour after the start of the test.

Blank

1. Answer the following True/False questions (20 points, 4 points for each correct answer, -2 points for each incorrect answer)

(1) The initializer list is an alternative syntax to setting up data members in the body of the constructor. They are equivalent in functionality

(2) The compiler provides your class with a built-in no-argument constructor, but once you define any constructor, the built-in one is no longer generated.

(3) If a function takes a parameter of type `const char*`, it can neither reassign nor delete the associated memory. For example, this won't work:

```
void foo(const char* inVal) {
  inVal = new const char[10]; // this won't compile
  delete[] inVal; // this won't compile
}
```

(4) To avoid memory leaks, every class should have a default constructor and destructor, even if the default, compiler-generated versions would suffice.

(5) Since references are really pointers, a reference can be re-bound to another object by taking that object's address. For example, if you start with:

```
int i, j;
int& intRef = i; // intRef is a reference to i
```

You can re-bind `intRef` to `j` by doing this:

```
intRef = &j; // intRef is now a reference to j
```

2. Write down the expected results of the following C++ programs. If it has a compile error or a runtime error, write down "compiler error" or "runtime error" with a brief explanation as an answer, respectively. (20 points)

(1) (5 points)

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

int *f (int n) {
    int a[10];
    for (int i = 0; i < 10; i++)
        a[i] = i*n;
    return a;
}

int main() {
    int j, k;
    int *p;
    j = 12;
    p = f(j);
    for (k = 0; k < 10; k++) {
        cout << p[k];
    }
    cout << endl;
    return 0;
}
```

(2) (5 points)

```
#include <iostream>
#include <string>
using namespace std;
int main() {
  string str = "We think in generalities, but we live in
details.";
  string str2 = str.substr(3, 5);
  size_t pos = str.find("in");
  string str3 = str.substr(pos);
  cout << str2 << "\n" << str3 << endl;
}
```

(3) (10 points)

```
#include <iostream>
using namespace std;
class A {
public:
    A() {
        cout << "A ctor" << endl;
    }
    ~A() {
        cout << "A dtor" << endl;
    }
    void foo() {
        cout << "A foo()" << endl;
    }
};

class B : public A {
public:
    B() {
        cout << "B ctor" << endl;
    }
    ~B() {
        cout << "B dtor" << endl;
    }
    void foo() {
        cout << "B foo()" << endl;
    }
};

A foo(A& input) {
    input.foo();
    return input;
}

int main() {
    B myB;
    B myOtherB;
    A myA;
    myOtherB = myB;
    myA = foo(myOtherB);
}
```

3. Write down the `swap()` functions so that the following code will show the execution results. (10 points)

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

int main() {
    string strA[] = {"aaa", "bbb", "ccc"};

    cout << strA[0] << " " << strA[1] << " " << strA[2] << endl;

    swap(&strA[1], &strA[2]);
    cout << strA[0] << " " << strA[1] << " " << strA[2] << endl;

    swap(strA[0], strA[2]);
    cout << strA[0] << " " << strA[1] << " " << strA[2] << endl;

    return 0;
}
```

Execution results:

aaa bbb ccc

aaa ccc bbb

bbb ccc aaa

4. Write a C++ program that takes three integer inputs, representing the side lengths of a triangle and prints out the type of a triangle. (20 points)

First, you should check that the sides are actually possible. This means that

- All three sides are positive
- Each side is shorter than the sum of the other two sides (the triangle inequality)

If it's not possible to build a triangle with those sides, print out **impossible** and nothing else. If it's possible to build a triangle with those sides, you will print out two facts about the triangle.

1. The first fact should be **right**, **acute**, or **obtuse**. In a **right** triangle, the longest side squared equals the sum of the squares of the other two sides (the Pythagorean theorem). If the longest side's square is larger than the sum of the other two sides, the triangle is **obtuse**. If smaller, it's called **acute**.
2. The second fact should be **equilateral**, **isosceles**, or **scalene**. In an **equilateral** triangle, all three sides are equal. In an **isosceles** triangle, exactly two sides are equal. In a **scalene** triangle, all sides have different lengths.

Execution results1: (bold represents a user input)
10 10 10☜
acute equilateral

Execution results2:
3 4 5☜
right scalene

Execution results3:
100 60 60☜
obtuse isosceles

5. Write down a C++ program that plays Rock, Paper, and Scissors game. (30 points)

(1) implement a class called `Tool`. It should have an `double` field called `strength` and a `char` field called `type`. You may make them either private or protected. The `Tool` class should also contain the function `void setStrength(double)`, which sets the strength for the `Tool`. (10 points)

(2) Create 3 more classes called `Rock`, `Paper`, and `Scissors`, which inherit from `Tool`. Each of these classes will need a constructor which will take in an `double` that is used to initialize the `strength` field. The constructor should also initialize the `type` field using `'r'` for Rock, `'p'` for Paper, and `'s'` for Scissors. (10 points)

(3) These classes also needs a public function `bool fight(Tool)` that compares their strengths in the following way: (10 points)

- Rock's strength is doubled (temporarily) when fighting scissors, but halved (temporarily) when fighting paper.
- In the same way, paper has the advantage against rock, and scissors against paper.
- The `strength` field shouldn't change in the function, which returns `true` if the original class wins in strength and `false` otherwise.

You may also include any extra auxiliary functions and/or fields in any of these classes.

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

class Tool {
    /* Fill in for (1) */
};

/*
    Implement class Scissors for (1) and (2)
*/

/*
    Implement class Paper for (1) and (2)
*/

/*
    Implement class Rock for (1) and (2)
*/

int main() {
    Scissors s1(5.0);
    Paper p1(7.0);
    Rock r1(15.0);
    cout << s1.fight(p1) << p1.fight(s1) << endl;
    cout << p1.fight(r1) << r1.fight(p1) << endl;
    cout << r1.fight(s1) << s1.fight(r1) << endl;

    return 0;
}
```