COMP 1006 – Introduction to Computer Science II
Quiz 1, Winter 2015

This is a closed book test. No calculators are allowed. All questions should be answered directly on this sheet in space provided. You have 45 minutes to complete this test. Assume that all C++ code (both given to you and that you write) is prefixed with the lines `#include <iostream>`, `#include <string>` and `using namespace std;`.

										TA Use Only
Name										
Student #	S	O	L	U	T	I	O	N	S	Grade (/20) TA

1. *(2 points)* What is the difference between the problem-solving techniques *Divide the problem* and *Reduce the problem*?

1 pt: divide
1 pt: reduce

- divide = break down into smaller pieces
- reduce = add/remove constraints to simplify problem

2. *(3 points)* In C++, explain what happens when you say `int a;`. Compare this to what happens when you say `new int;`.

1 pt: explain int a;
1 pt: explain new int;
1 pt: indicate difference (stack/heap or compile/run)

int a; → get aside space on stack at compile time for an int

new int; → get aside space on heap at run time for an int, return a pointer to it

For the next two questions, assume that the following **struct** has been defined:

```
struct zergling {
    int health;
    int maximum_health;
};
```

3. *(3 points)* What is the output of the following C++ code? Why?

```
void healZergling(zergling z) {
    z.health += 5;
    if(z.health > z.maximum_health) {
        z.health = z.maximum_health;
    }
}

int main() {
    zergling joe = { 20, 45 };
    cout << joe.health << endl;
    healZergling(joe);
    cout << joe.health << endl;
    return(0);
}
```

Output: 20
20

1 pt

Reason: joe is passed by value and so only the health of the copy is changed.

1 pt: "pass by value"
1 pt: "does not change joe"

4. *(4 points)* The following C++ code compiles without any errors, but there are two problems that occur at run-time. Identify and explain both of these problems.

```
int main() {
    zergling army[3];
    for(int i = 0; i < 3; i++) {
       army[i].health = 45;
    }
    for(int i = 0; i < 4; i++) {
        if(army[i].health < army[i].maximum_health) {
            cout << "Zergling " << i << " is damaged!" << endl;
        }
    }
}
```

1 pt: identify
1 pt: explain

1. Second for-loop goes to i=3, which is past the end of the array. This could cause garbage to be compared in the if-statement, and so the output is not predictable in the last iteration.
2. The first for-loop never initializes the maximum_health member, and so it will contain garbage.

5. *(8 points)* In this question, you will have to write some C++ code. It does not have to be perfect: missing a semicolon (for example) is fine. However, given the code you write here, it should be very straightforward to get it to compile with minimal effort.

1 pt (a) Define a **`struct`** that represents a chocolate bar. It should store the brand, the weight (in grams), and the *deliciousness* on a scale from 1-10. It is up to you to determine what types to use for each of these.

2 pts (b) Create (on the stack) an array that consists of three chocolate bar **`struct`**s. Initialize each chocolate bar using values of your choice.

2 pts (c) Write a function that takes an array of chocolate bars and returns the name of chocolate bar with maximum deliciousness. The function should not change the array or its contents.

2 pts (d) Write a function that takes a single chocolate bar and increase its deliciousness by one. The function should not return anything.

1 pt (e) Call each of these functions once from `main()`.

(a)
```
struct chocolate {
    string brand;
    int weight;
    int deliciousness;
};
```
OK if float

(b)
```
chocolate chocolates[3];
chocolates[0].brand = "Slcor";
chocolates[0].weight = 39;
chocolates[0].deliciousness = 9;
(etc.)
```
assume this is in main

(c)
```
string best(chocolate chocolates[],
            int size) {
  int best_index = 0;
  for(int i=0; i<size; i++){
    if (chocolates[i].deliciousness
      > chocolates[best_index]
        .deliciousness) {
      best_index = i;
    }
  }
  return chocolates[best_index]
      .name;
}
```
-1 if missing

(d)
```
void increase(chocolate &c) {
  c.deliciousness++;
}
```
OK to pass by pointer, but -1 if just passing a struct by value.

(e)
```
cout << best(chocolates, 3)
     << endl;
cout << chocolates[0].
        deliciousness
     << endl;
increase(chocolates[0]);
cout << chocolates[0].
        deliciousness
     << endl;
```
Doesn't matter what is done, as long as calls are made.