Final Exam
CS253 Fall 2014

Your Name (Print): _____

Instructions:

For this exam, like the midterm, you will get a packet of code plus this set of questions. There are a total of 25 questions. Every question is worth 4 points.

This exam is closed book, closed note, and closed neighbor. All work must be your own. You have two hours.

Warning: some of the questions on this exam are subtle; others are outright hard. Take your time, and if you have time at the end look over your answers. Whatever you do, don't panic – this test will be graded on a curve.

Part 1: Questions pertaining to file Image.tpp

1. The last line in the file Image.tpp is `#include <window.tpp>.` Why isn't this line at the top of the file? Explain.

2. The Inside() method of the Image class is a Boolean method that returns true if a (row, col) coordinate is inside the image, and false if it is out of bounds. The Inside method explicitly tests if row or col is too large. Why doesn't it test if they are too small?

3. The templated image class in file Image.tpp defines the iterator type for images. To extend the STL it should also define the reference type. Write the line of code that should be added to the Image class definition to do this.

4. Assume a program has the following line (and template.cpp has been included):

   ```
   Image<double> foo;
   ```

   What is the value of `foo.data` after this line is executed?

Questions 5-7 assume that an object called `img` of type `Image<int>` already exists in a program:

5. Consider the following line of code:

```
Window<int> w = img.GetWindow(0,0,1,1);
```

How many int and how many unsigned int are allocated on the stack as a result of this line of code?

6. Considering the same line of code as above (Question #5), how many int* are allocated on the stack?

7. Considering the same line of code as in Question #5, how many int* are allocated on the heap?

8. Assume a method of the Image class has the following line:

```
unsigned int bar = row_ptr[1] - data;
```

What is the value of bar after the line is executed?

9. Consider the following line of code:

```
Image<Image<int> > foo(3,5);
```

Is this legal? (i.e. will it compile without error?) If it is, how many int* are allocated on the heap as a result of this line of code?

10. Below are four possible signatures (declarations) of the Set method inside the Image class. Assume the body of the Set method does not change, only the signatures. Which ones will generate compile time errors? For the ones that generate errors, explain why.

```
(a) void Image<PIXEL>::Set(PIXEL value)
(b) void Image<PIXEL>::Set(PIXEL value) const
(c) void Image<PIXEL>::Set(const PIXEL value)
(d) void Image<PIXEL>::Set(const PIXEL& value)
(e) void Image<PIXEL>::Set(const PIXEL& value) const
```

11. Of the options in question #10 above, which would you recommend and why?

For questions 12 & 13, assume the following class definition:

```
class RGB {
public:
   RGB(int r, int g, int b) : red(r), green(g), blue(b) {}
   RGB(int val) : red(val), green(val), blue(val) {}
   ~RGB() {red = 0; green = 0; blue = 0;}
   int red;
   int green;
   int blue;
};
```

12. Add a method definition to the RGB class that makes the following legal:

```
Image<int> i_image(2,2,1);
Image<RGB> c_image(2,2,RGB(1,2,3));
i_image.Copy(c_image);
```

Note: you do not have to implement the method, just define its signature.

13. Using the same RGB class as above, how many times is the RGB destructor called as a result of the following code?

```
Image<int>* img_ptr = new Image<int>(2,1);
delete img_ptr;
```

For the remaining questions, assume that main.cpp is compiled and that the executable program is called *exam.*

14. If the program is executed from the command line as:

    ```
    ./exam 0
    ```

    How many times is an Image constructor (any of them) executed? How many times is the Image destructor executed?

15. What does the program print when invoked as :

    ```
    ./exam 0
    ```

    If it crashes or throws an exception, say so.

16. True or False: valgrind would detect a memory error (e.g. uninitialized value, out of bounds reference) when the program is invoked as:

    ```
    valgrind ./exam 0.
    ```

17. Assume the program is invoked from the command line as:

```
./exam 1
```

How many times is a Window constructor (any of them) executed? How many times is the Window destructor executed?

18. What does the program print when invoked as:

```
./exam 1
```

If it crashes or throws an exception, say so.

19. True or False: valgrind will detect a memory error (e.g. uninitialized value, out of bounds reference) when the program is invoked as:

```
valgrind ./exam 1
```

20. Assume the program is executed from the command line as `./exam 2`

    How many times is a Window constructor (any of them) executed? How many times is the Window destructor executed?

21. What does the program print when invoked as
    `./exam 2`

    If it crashes or throws an exception, say so.

22. True or False: valgrind would detect a memory error (e.g. uninitialized value, out of bounds reference) when the program is invoked as:

    valgrind `./exam 2`

23. How many bytes of memory are allocated on the heap when the program is invoked as:

```
./exam 2
```

24. The following code does not compile. Explain why.

```
Image<int> img(2,2,0);
for(Image<int>::iterator iter = img.begin();
    iter != img.end();
    iter++)
  {}
```

25. As written, does the Image<PIXEL>::iterator class iterate over the 2D image array in row major order or column major order? (In other words, does it access all the pixels in the first row, followed by all the pixels in the second, and so on, or does it access all the pixels in the first column, followed by all the pixels in the second column, and so on.)