ECE 551D
Fall 2017 Practice Final Exam

Name: NetID:

There are 9 questions, with the point values as shown below. You have 175 minutes with a total of 115 points. Pace yourself accordingly.

This exam must be individual work. You may not collaborate with your fellow students. However, this exam is open notes, so you may use your class notes, which must be handwritten by you.

I certify that the work shown on this exam is my own work, and that I have neither given nor received improper assistance of any form in the completion of this work.

Signature:

#	Question	Points Earned	Points Possible
1	Multiple Choice		20
2	Data Structure Concepts		12
3	Reading Code 1		8
4	Reading Code 2		8
5	Reading Code Concurrency		8
6	Coding 1		12
7	Coding 2		12
8	Coding 3		16
9	Coding 4		19
	Total		115
	Percent		100

Question 1 Multiple Choice [20 pts]

1. Debugging is best performed by
   (a) Guess and check.
   (b) Applying the scientific method.
   (c) Using Pascal's Second Theorem.
   (d) Converting all pointers to references.
   (e) None of the above

2. What is the effect of declaring a method as `virtual`?
   (a) It does not require an entry in the vtable.
   (b) The actual instructions to implement it are placed inside the objects.
   (c) The method can still be used after an object is destroyed.
   (d) The method will be dynamically dispatched.
   (e) None of the above

3. Why do you write the definition of a templated class in a header file, rather than just the interface?
   (a) Template definitions must be visible to the compiler at instantiation.
   (b) Template arguments must be compile-time constants.
   (c) Templates are only type-checked when specialized.
   (d) You can provide an explicit specialization for a template.
   (e) None of the above

4. Temp-and-swap is a common idiom for writing code with which level of exception safety?
   (a) No-throw Guarantee
   (b) Strong Guarantee
   (c) Basic Guarantee
   (d) Velociraptor-free Guarantee
   (e) No Guarantee

5. What is the typical (*not* worst case) access time for a binary search tree?
   (a) $O(1)$
   (b) $O(\lg(N))$
   (c) $O(N)$
   (d) $O(N^2)$
   (e) None of the above

6. What is the primary advantage of merge sort over quick sort?
   (a) Merge sort has guaranteed $O(N \lg(N))$, and quick sort does not.
   (b) Quick sort needs extra space, and merge sort does not.
   (c) Merge sort has good locality and quick sort does not.
   (d) Merge sort is stable, and quick sort is not.
   (e) None of the above

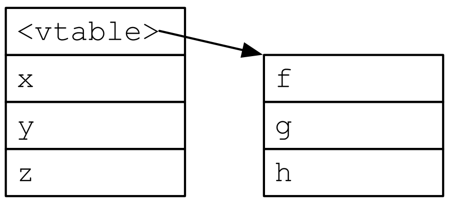
7. When using multiple threads, a programmer passes a function pointer to `pthread_create` to specify
   (a) What to do in the event of an error.
   (b) How to deal with un-handled C++ exceptions on that thread.
   (c) What to do in the case of deadlock.
   (d) The default properties of locks.
   (e) None of the above

8. Which parallel programming idiom is useful in divide-and-conquer algorithms?
   (a) Data parallelism
   (b) Pipeline parallelism
   (c) Task parallelism
   (d) Embarrassing parallelism
   (e) None of the above

9. If we have the following classes:

```
class A {
  int x;
  virtual void f();
};
class B {
  int y;
  virtual void g();
};
class C : public A, public B {
  int z;
  virtual void h();
};
```

Which best explains why the following object layout is incorrect?



(a) Function `g` should be before `f` in the vtable.
(b) The vtable pointer must be adjusted.
(c) It violates the subobject rule.
(d) It violates the open/closed principle.
(e) None of the above

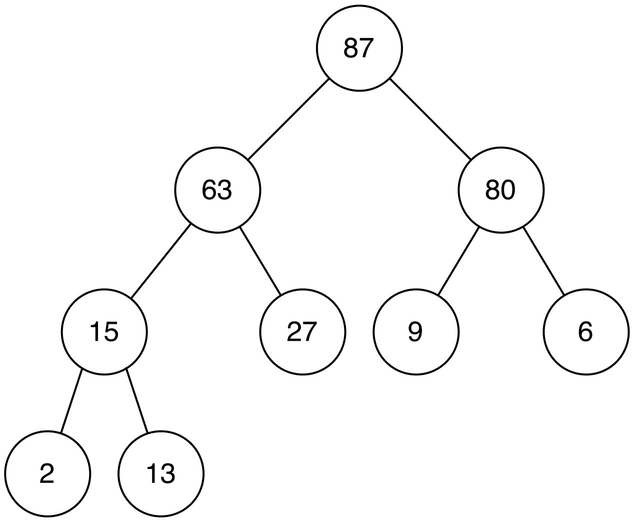
10. If you use virtual inheritance, there are performance overheads for accessing

(a) All fields.
(b) Fields in the virtually inherited parent.
(c) Fields inherited through non-virtual parents.
(d) No fields.
(e) None of the above (some other combination)

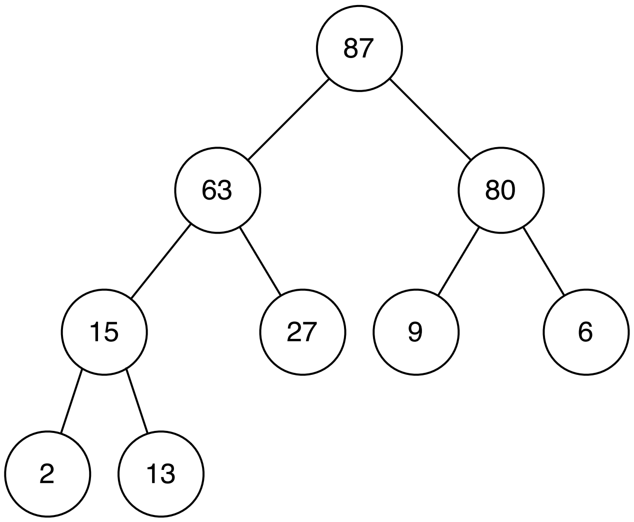
Question 2 Data Structure Concepts [12 pts]

Show the results of performing each of the following operations on the given data structures:

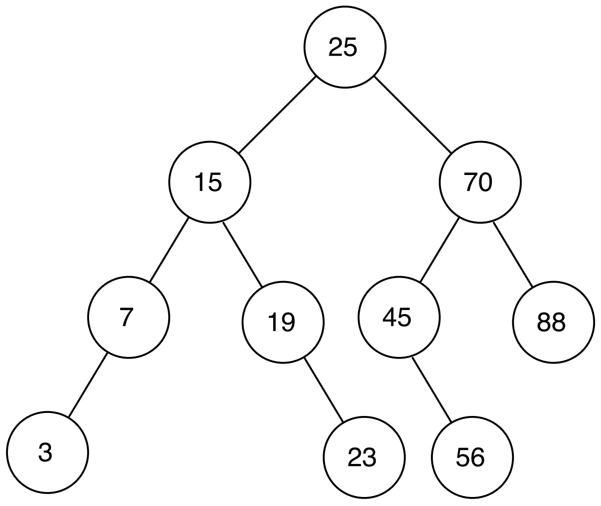
1. Add 94 to the following max-heap



2. Remove the maximum element from the following max-heap:



3. Add 4, then add 51 to the following (regular, un-balanced) BST:



4. Remove 25 from the BST in the previous part (before you added 4 and 51):

Question 3 Reading Code 1 [8 pts]

What is the output when the following C++ code is executed?

```
#include <iostream>
#include <cstdlib>

class X {
public:
  int y;
  X(): y(7) {
    std::cout << "X()\n";
  }
  X(int z): y(z) {
    std::cout << "X(" << z << ")\n";
  }
  ~X() {
    std::cout << "~X(" << y << ")\n";
  }
  X & operator+=(const X & rhs) {
    std::cout << "Doing " << y << " += " << rhs.y << "\n";
    y += rhs.y;
    return *this;
  }
};
X f(X & y, X z) {
  X ans(y);
  ans += z;
  y += ans;
  z += ans;
  return ans;
}
int main(void) {
  X a;
  X b(3);
  X c = f(a, b);
  std::cout << a.y << "\n";
  std::cout << b.y << "\n";
  std::cout << c.y << "\n";
  return EXIT_SUCCESS;
}
```

Question 4 Reading Code 2 [8 pts]

What is the output when the following C++ code is executed?

```
#include <cstdlib>
#include <iostream>
#include <vector>
class CurrencyData {
protected:
  double dollar_ex_rate;
public:
  explicit CurrencyData(double ex_rate) : dollar_ex_rate(ex_rate) {
    std::cout << "CurrencyData(" << ex_rate << ")\n";
  }
  virtual ~CurrencyData() { std::cout << "~CurrencyData()\n"; };
  virtual void printDollarAmt(double money) = 0;
};
class Euro : public CurrencyData {
public:
  Euro() : CurrencyData(1.1) { std::cout << "Euro()\n"; }
  virtual ~Euro() { std::cout << "~Euro()\n"; };
  virtual void printDollarAmt(double money) {
    std::cout << money << " Euro = " << money * dollar_ex_rate << " Dollars\n";
  }
};
class Yuan : public CurrencyData {
public:
  Yuan() : CurrencyData(0.15) { std::cout << "Yuan()\n"; }
  virtual ~Yuan() { std::cout << "~Yuan()\n"; };
  virtual void printDollarAmt(double money) {
    std::cout << money << " Yuan = " << money * dollar_ex_rate << " Dollars\n";
  }
};
int main(void) {
  std::vector<CurrencyData *> vec;
  vec.push_back(new Euro());
  vec.push_back(new Yuan());
  double amt = 10;
  for (std::vector<CurrencyData*>::iterator it=vec.begin(); it!=vec.end(); ++it) {
    (*it)->printDollarAmt(amt);
    amt *= 10;
    delete *it;
  }
  return EXIT_SUCCESS;
}
```

Question 5 Reading Code Concurrency [8 pts]

Consider the following multi-threaded code (which has multiple possible outputs):

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

pthread_mutex_t lock;

struct point_tag {
  int x;
  int y;
  int z;
};
typedef struct point_tag point_t;

void * f (void * p_in) {
  point_t * p = p_in;
  p->x = 1;
  pthread_mutex_lock(&lock);
  p->y = 2;
  p->z = 3;
  pthread_mutex_unlock(&lock);
  pthread_mutex_lock(&lock);
  printf("In f: %4d %d %d\n", p->x, p->y, p->z);
  pthread_mutex_unlock(&lock);
  return NULL;
}

int main(void) {
  point_t pt = {.x = 0, .y = 0, .z = 0};
  pthread_t thr;
  pthread_create(&thr, NULL, f, &pt);
  pt.x = 4;
  pthread_mutex_lock(&lock);
  pt.y = 5;
  pthread_mutex_unlock(&lock);
  pthread_join(thr, NULL);
  pt.z = 6;
  printf("In main: %d %d %d\n", pt.x, pt.y, pt.z);
  return EXIT_SUCCESS;
}
```

For each of the following outputs, select “Possible” if the program could generate that output or select “Impossible” if the program could not generate that output.

1.
   ```
   In f:    1 2 3
   In main: 4 5 6
   ```
   The above output is: **Possible** **Impossible**

2.
   ```
   In f:    1 2 3
   In main: 4 5 3
   ```
   The above output is: **Possible** **Impossible**

3.
   ```
   In f:    1 5 3
   In main: 1 5 6
   ```
   The above output is: **Possible** **Impossible**

4.
   ```
   In f:    4 5 3
   In main: 4 5 6
   ```
   The above output is: **Possible** **Impossible**

5.
   ```
   In f:    4 5 3
   In main: 1 5 6
   ```
   The above output is: **Possible** **Impossible**

Question 6 Coding 1 [12 pts]

Declare and write the templated function `addToAll`, which is templated over one type parameter, `T`. This function should take two parameters: (1) a const reference to a `std::set s` and (2) an `int n`. This function should create and return a set that is just like `s`, except that each element has had `n` added to it. You may assume that this function is only used on types such that `operator+` is defined on `(T, int)`.

Question 7 Coding 2 [12 pts]

Declare and write the templated function `remapSet`, which is templated over two type parameters, `T` and `S`. This function should take two parameters: (1) A const reference to a `std::set` of `T`s. (2) A const reference to a `std::map` which maps `T`s to `S`es. This function should return a `std::set` of `S`es.
In particular, this function should construct its answer set by taking each item in the input set, using it as a key in the input map, and adding the resulting value from the map to the answer set. If the input map does not contain a mapping for an item in the input set, your function should throw a `std::domain_error` exception.

Question 8 Coding 3 [16 pts]

Write the `removeGreater` function in the following `LinkedList` class (which only holds `ints`). This function should remove *all* of the elements of the list which are greater than the `threshold` parameter.

```
class LinkedList {
private:
  class Node{
  public:
    int data;
    Node * next;
    Node(int data_): next(NULL), data(data_) {}
    Node(int data_, Node * next_): next(next_), data(data_) {}
  };
  Node * head;
public:

  void removeGreater(int threshold) {









  }
};
```

Question 9 Coding 4 [19 pts]

Suppose you are testing and debugging the following `RedBlackTree` class (which holds `ints`):

```
class RedBlackTree {
private:
  class Node {
  public:
    bool isBlack;  //true means this node is black, false means red
    int data;
    Node * left;
    Node * right;
  };
  Node * root;

  void checkBlackCounts() const {
    //you will write this
  }

public:
  //constructors, destructors, other methods not shown
};
```

As part of your testing and debugging of this `RedBlackTree` class you have decided to write a method to validate that your coloring is correct. As part of this validation, you have decided to write a method `checkBlackCounts`, which checks that this tree obeys the rule that every path from the root to NULL has the same number of black nodes. For **full credit, your code should be const-correct** (meaning you write const in exactly the places where you should use it). You should use `assert` in your solution, such that if the tree is incorrect, an `assert` will fail. If the tree is correct, your method should return normally.
You do NOT need to write any code to validate the other red/black tree rules—only the one described above.

Answer on Next Page

Write your code here, along with any helper methods (all of which we will assume are members of the `RedBlackTree` class):