Name: ______________________________________

USC NetID (e.g., `ttrojan`): ____________________________

CS 455 Final Exam Spring 2017 [Bono]

May 10, 2017

There are 7 problems on the exam, with 74 points total available. There are 12 pages to the exam (6 pages double-sided), including this one; make sure you have all of them. There is also a one-page code handout that accompanies the exam. If you need additional space to write any answers or for scratch work, pages 10, 11, and 12 of the exam are left blank for that purpose. If you use these pages for answers you just need to direct us to look there. ***Do not detach any pages from this exam.***

Note: if you give multiple answers for a problem, we will only grade the first one. Avoid this issue by labeling and **circling** your final answers and crossing out any other answers you changed your mind about (though it's fine if you show your work).

Put your name and USC username (a.k.a., NetID) at the top of the exam. Also, put your NetID at the top right of the front side of each page of the exam. Please read over the whole test before beginning. Good luck!

Problem 1 [10 pts]

[Java] Briefly describe two possible internal representations one could use in an implementation of the Java `Map` interface, such that each of the representations is *different* than the two implementations that are provided in the Java library that we discussed this semester. For each one you describe, give the worst-case big-O time to do each of the `Map` operations listed below.

```
  // get the value associated with this key
  // if no entry with this key exists, returns null
  public ValueType get(KeyType key)

  // inserts an entry or updates the value that goes with an existing
  // key.  Returns the old value associated with the key, or null
  // if this key was not previously in the map
  public ValueType put(KeyType key, ValueType value)

  // remove the entry with this key
  // returns the associated value or null if not found
  public ValueType remove(KeyType key)
```

Description of representation 1:

Big-O times for representation 1:

`_________get          _________put           _________remove`

Description of representation 2:

Big-O times for representation 2:

`_________get          _________put           _________remove`

Problem 2 [8 pts.]

[Java] Implement the `boolean` Java method `sortByScore`, which sorts an `ArrayList` of `Student's` so they are in decreasing order by score. For students with the same score, they must be in increasing order by name. ***You must use the Java sort utility*** (more info about that on the code handout), and include any additional code necessary to make it work (outside of the `sortByScore` method).

Here is an example of contents of such an `ArrayList` shown in the order it would be in after a call to `sortByScore`:

Jan 98
Lin 98
Ann 84
Fred 84
Moe 84
Zhou 80
Aarti 72

Here is the `Student` class it uses (only shows interface, since that's all we are using here). This is an already completed class; ***you may not modify `Student` for this problem***.

```
public class Student {
   public Student(String name, int score) { . . . }
   public String getName() { . . . }
   public int getScore() { . . . }
   public void changeScore(int newScore) { . . . }

   . . .  // private stuff not shown
}


public static void sortByScore(ArrayList<Student> students) {
```

Problem 3 [19 pts. total]

Part A (15). [Java] Write the complete class definition for a *Java* class for a `Stack` of `int`s that uses an array representation (not `ArrayList`) such that all stack operations run in amortized constant time or better. Hint: Use the strategy such that if you run out of space in the array, you double its size. (The "amortized" part of the time is because of the need to increase the array size periodically.) You may not use any Java library classes or methods in your code, except the the `Arrays.copyOf` method, described on the code handout. You do not have to write method comments for your class.

The complete interface for the `Stack` class for this problem is shown by example below (Note: method names shown in bold):

```
Stack s = new Stack();            // creates an empty stack

s.push(3);          // pushes 3 onto the stack

s.push(0);          // pushes 0 onto the stack

int val1 = s.pop();   // pops top element from the stack and returns its value.
                      // only legal on non-empty stack

int val = s.top();    // returns the top element (stack is unchanged)
                      // only legal on non-empty stack

if (s.isEmpty()) {  // returns true if stack is empty, otherwise false

  System.out.println("stack is empty.");

}
```

Space for your answer is given on this and the next page.

Problem 3 (cont.)

(additional space for your answer to Part A; Part B)

Part B (4). Write a representation invariant for your class implementation:

Problem 4 [9 points]

[C++] Consider the following buggy C++ function, `insertFront`, that attempts to insert a value in the front of a linked list, and the program that uses it below. (See code handout for documentation of `Node` and `ListType`):

```
// printList: prints list using the format: (cab sit dog dad)
// prints an empty list as: ()
// PRE: list is a well-formed list
void printList(ListType list) {  /* code not shown */  }

void insertFront(ListType & list, const string & val) {

   Node * newGuy = new Node(val);
   list->next = newGuy;

}

int main() {

   ListType myList = new Node("a");
   insertFront(myList, "b");
   insertFront(myList, "c");  // ***

   insertFront(myList, "d");

   printList(myList);

}
```

Part A. In the space above, draw a box-and-pointer diagram (a.k.a., memory diagram) showing all variables, objects, and their state as they have changed during the code sequence up *through* the statement labeled `***` (do not show the stuff after that). This includes showing `insertFront`'s parameters (you can name them $\texttt{list}_1$, $\texttt{val}_1$, in the first call and $\texttt{list}_2$, $\texttt{val}_2$, in the second call).

Part B. What is the output of the program?

Problem 5 [8 points]

[C++] The following is a simplified version of the `freq.cpp` program we did in lecture, but that is now organized as a multi-file program, so that we can compile the histogram functions (using `histogram.h` and `histogram.cpp`) separately from this program that uses them (`freq.cpp`). **In the space provided below show what needs to be in `histogram.h`** (in the blank box at the bottom of the page). **You may not change the contents of the other two files.** A full credit answer has what's needed and nothing more. Note: all places where functions from this program are called are shown in the code. Note: no "#fndef", etc., needed, because no classes are defined

```
#include "histogram.h"
const int MAX = 10;
const int NUM_COUNTS = MAX + 1;

int main() {
   int counts[NUM_COUNTS];
   initCounts(counts,  NUM_COUNTS);
   readAndCompute(counts,  NUM_COUNTS);
   printHist(counts,  NUM_COUNTS);
}
```

freq.cpp

```
#include "histogram.h"
#include <iostream>
using namespace std;
// Some functions definitions are elided for brevity
void initCounts(int counts[], int size) { . . . }
void readAndCompute(int counts[], int size) { . . . }
void printTicks(int numTicks) { . . . }
void printHist(int counts[], int size) {
   . . .
   printTicks(…);
   . . .
}
```

histogram.cpp

histogram.h

Problem 6 [5 pts]

[C++] Consider the following class and in particular the **`add`** method that attempts to insert a value into a hash table that uses chaining. Note: you may assume that the hash function used, `hashCode`, and the helper function, `insertFront` (described below), both work correctly:

```
// inserts elmt in the front of list
// pre: list is a well-formed list
// Note: assume this insertFront works correctly
void insertFront(ListType &list, const string & elmt);


class HashSetOfStrings {

  public:

    HashSetOfStrings();  // creates an empty set

    //  Add elmt to the set if it is not already present.
    //  If elmt is already present returns false and no change is made to the set.
    bool add(const string & elmt);

    // . . . [other methods of HashSetOfStrings not shown]

  private:

   const int HASH_SIZE = 9973;
   ListType hashArray[HASH_SIZE];          // fixed size array in this version

   // returns a value in the range 0 through HASH_SIZE-1
   int hashCode(const string & elmt) const;

   // . . . other details of class left out

 };


bool HashSetOfStrings::add(const string & elmt) {

        int hashIndex = hashCode(elmt);

        if (hashArray[hashIndex] != NULL) { return false; }

        insertFront(hashArray[hashIndex], elmt);

        return true;

}
```

Give a concrete example that illustrates a scenario where the `add` code does not work. Your answer may consist of some combination a description of the scenario, code, and an explanation of what happens that is incorrect.

Problem 7 [15 points]

[C++] Write the C++ function `gut`, which removes all of the middle elements from a linked list. That is, it keeps only the first and last elements of the list. For full credit, your function must reclaim any dynamic memory no longer used by the list.

The `Node` and `ListType` definitions are on the code handout. Some examples:

`list`	**`list`** after call to **`gut(list)`**
`(the big sloppy dog)`	`(the dog)`
`(the dog)`	`(the dog)`
`(the)`	`(the)`
`(the wet dog)`	`(the dog)`
`()`	`()`

```
// removes all but the first and last element from the list
// PRE: list a well-formed linked list

void gut(ListType & list) {
```

Extra space for answers or scratch work. (DO NOT detach this page.)

If you put any of your answers here, please write a note on the question page directing us to look here. Also label any such answers here with the question number and part, and circle the answer.

Extra space for answers or scratch work (cont.) (DO NOT detach this page.)

If you put any of your answers here, please write a note on the question page directing us to look here. Also label any such answers here with the question number and part, and circle the answer.

Extra space for answers or scratch work (cont.) (DO NOT detach this page.)

If you put any of your answers here, please write a note on the question page directing us to look here. Also label any such answers here with the question number and part, and circle the answer.