CS302 Midterm Exam. October 20, 2011. James S. Plank

Answer all questions. Write your answers on the answer sheets provided.

Question 1

Describe the disjoint set data structure. In particular, do the following:

- Give a one paragraph overview of the data structure.
- Give me a reasonable API in C++. The API should include public methods and protected data.
- For each method in the API, describe what the method does (logically, not its implementation).
- For each method in the API, state its running time.
- In class, we used disjoint sets to generate rectangular mazes that had a maximum number of walls between cells. Explain how disjoint sets solve this problem.

Question 2

Suppose you do not have access to a sorting library. Which sorting algorithm would you implement for each of the following sorting problems, and why?

1. You are sorting 1,000,000 elements of totally random data, where you know nothing about it.
2. You are sorting 1,000,000 random doubles uniformly distributed between 500 and 1000.
3. You are sorting a vector with 10 random doubles. (And suppose you'll be doing this a million times).
4. You are sorting the output of the program to the right.

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

main()
{
  double i;

  srand48(time(0));
  for (i = 0; i < 1000000; i++) {
    printf("%.4lf\n", i + drand48()*5);
  }
  exit(0);
}
```

Question 3

Your boss is a well-meaning, but kind of technologically lame geologist. She has a data file where each line is of the form:

```
Test-Name Number-of-runs Avg-result
```

The name is a string without spaces. The number of runs is an integer, and the avg-result is a double. A small example file is shown below:

```
BORING-MINERAL      4  45.353
DULL-ROCK           9  59.408
BORING-MINERAL      5  41.327
DULL-ROCK           1  77.155
SOPORIFIC-FOSSIL    5  15.116
```

As you can see, some tests have multiple lines, because they are run at different times. Your boss would like to see the total number of runs for each test, and she's been trying to get excel to do it, but that is leading to frustration. She wants you to write a simple program that reads the input file, then outputs each test preceded by its total number of runs. The output should be sorted from most runs to least. If two tests have the same number of runs, don't worry about their relative order. Print the number of tests right-justified, padded to four characters. For example, on the input to the right, you'll get the following output:

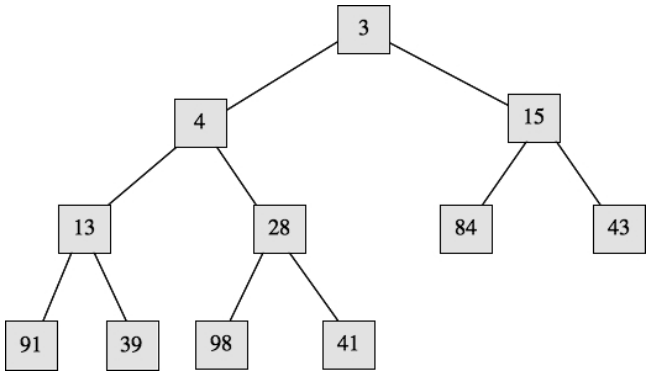
```
  10 DULL-ROCK
   9 BORING-MINERAL
   5 SOPORIFIC-FOSSIL
```

Question 4

Part A: Draw the vector representation of the heap to the right.

Part B: Draw the heap that results when when you call **Push(2)** on the heap to the right.

Part C: Draw the heap that results when when you call **Pop()** on the heap to the right. **Do not call Pop() on your answer to Part B -- call it on the heap to the right**.



Part D: Suppose one calls the priority queue constructor with the vector below:

29	24	87	95	69	60	99	32	31	25	23	71	70	75	68

There are two ways that we learned to implement this constructor. The first simply calls **Push()** on each element. I want you to draw me the heap that results when you implement it in the second, more efficient way. Don't bother drawing the vector representation -- draw the graphical representation.

Part E: Suppose a heap contains *n* elements. What is the running time (big-O) of **Push()**?

Part F: Suppose a heap contains *n* elements. What is the running time (big-O) of **Pop()**?

Part G: What is the running time (big-O) of constructing a heap from a vector with *n* elements as in **Part D**?

Question 5

Write the procedure **print_subset()**, which has the following prototype:

```
void print_subset(int subset, vector <string> &names);
```

The vector **names** has fewer than 31 elements. Thus, we can represent any subset of **names** with an integer, as described in class. Write **print_subset()**, which prints the elements of **names** that are in the specified integer **subset**. You'll have to use bit arithmetic. Just print one name per line.