CS302 2012 Midterm Exam. James S. Plank. October 9, 2012. Answer all questions

Question 1

To the right, I give you six vector representations of heaps, each with an action. Your job is to give me the vector representation of the heap that results when the action has been performed. Put the answers on the answer sheet for question one. I have also given you some scratch sheets that will help you do this problem more quickly.

#	Heap	Action
1	11 16 14 18 36 65 70 53	Push 82
2	6 25 41 68 56 94 48	Pop
3	5 10 38 49 26 93 63 98 65 71 67	Pop
4	33 36 35 58 65 68 65 100 90	Push 21
5	18 79 52 98 91 83 86	Push 1
6	1 11 16 18 50 53 100 84 49 75	Push 15

Question 2

Recall the header file for disjoint sets, listed to the right. I have a program which creates an instance of **Disjoint** in the parameter **dj**, then does **Union()** and **Find()** operations to reach a point where it executes the following statements:

```
class Disjoint {
  public:
    Disjoint(int nelements);
    int Union(int s1, int s2);
    int Find(int element);
    void Print();
  protected:
    vector <int> links;
    vector <int> ranks;
};
```

```
dj.Print();
s3 = dj.Union(s1, s2);
printf("Union(%d,%d) = %d\n", s1, s2, s3);
printf("Find(%d) = %d\n", e1, dj.Find(e1));
printf("Find(%d) = %d\n", e2, dj.Find(e2));
dj.Print();
```

On the answer sheet, there are four example outputs to the statements above. For each output, the implementation of **Disjoint** is stated. Your job is to fill in the missing output. To be specific, you'll see the output of the first **Print()** statement, and then you'll get to see the values of **s1**, **s2**, **e1** and **e2**. Your job is to print out the return values of the **Union()** and the two **Find()** calls, and then to show the output of the last **Print()** statement. In **Union()**, break ties so that the higher set id is the parent of the lower.

*To make your life easier, you only need to show the values in the **Print()** statement that have changed from the previous values.*

Question 3: For each action below, state the running time in terms of the most precise big-O notation. Assume that the data structures have n elements. To help you out, each answer will be one of the following: $O(1)$, $O(n)$, $O(\log n)$, $O(n \log n)$, $O(n^2)$, $O(2^n)$, $O(\alpha(n))$.

A: Sorting a vector using insertion sort.
B: Sorting a vector using heap sort.
C: Calling **Push()** on a heap.
D: Creating a heap from a random vector.
E: Calling **Pop()** on a heap.
F: Printing all elements of a map in order.
G: Printing all subsets of a set.
H: Calling **Union()** on a disjoint set.
I: Deleting an element from a map.
J: Printing all pairs of elements in a vector of integers.
K: Inserting an element into a map.
L: Sorting a vector using selection sort.
M: Sorting a vector using bubble sort.
N: Calling **Find()** on a disjoint set.
O: Sorting a vector using STL multisets.
P: Enumerating all 2-disk failures in an n-disk system.