CSCI 480 - APS: Sample Exam 

Duration: 65 minutes

Name:_____

NetID:_____

Student to your left: **Student to your right:**

_____ _____

DO NOT OPEN THIS EXAM UNTIL INSTRUCTED

Instructions:

- **Write your full name and your NetID on the front of this exam.**
- Write the names of students sitting to your left and right.
- Make sure that your exam is not missing any sheets. There should be six (6) double sided pages in the exam.
- Write your answers in the space provided below each problem. If you make a mess, clearly indicate your final answer (or state that it is on the scrap paper).
- If you have any questions during the exam, raise your hand and we will get to you.
- At the end of the exam, there is one blank page. Use this as your scrap paper. If you need additional scrap paper, raise your hand and we will get it for you.
- This exam is closed books, closed notes, closed computers. You are allowed one double sided page of your own notes.
- **You need to stay in your seat until the exam is finished.** You should not leave the room even if you finish the exam. This distracts other students who are still working.

Good luck!

Joanna Klukowska
joannakl@cs.nyu.edu

Problem 1 (? points).

Write an efficient algorithm that produces a list of all prime numbers smaller than $1,000,000$.

Joanna Klukowska
joannakl@cs.nyu.edu

Problem 2 (? points).

A typical machine executes approximately 10^9 operations per second (on average, since some operations are slower than others).

Assume you have three algorithms with different performance for the same problem: $O(N^2 \log N)$, $O(N \log N)$, $O(N)$. For each of those algorithms determine if it will complete in under 2 seconds when executed on the input size of $N = 100,000$.

Show your calculations or explain how you got your answer.

A. $O(N^2 \log N)$ algorithm

B. $O(N \log N)$ algorithm

C. $O(N)$ algorithm

Joanna Klukowska
joannakl@cs.nyu.edu

Problem 3 (? points).

The following is an algorithm for converting infix expressions to postfix expressions.

```
for each tokn in the input infix string expression
    if the token is an operand
        append to postfix string expression
    else if the token is a left brace
        push it onto the operator stack
    else if the token is an operator
        if the stack is not empty
            while top element on the stack is not a left brace AND has higher or equal precedence
                pop the stack and append to postfix string expression
        push it (the current operator) onto the operator stack
    else if the token is a right brace
        while the operator stack is not empty
            if the top of the operator stack is not a matching left brace
                pop the operator stack and append to postfix string expression
            else
                pop the left brace and discard
                break
while the operator stack is not empty
    pop the operator stack and append to postfix string expression
```

Apply the infix to postfix conversion to the following expression:

$$5 * (6 + 4)/(8 + 2) - 7$$

Show the content of the operator stack and the postfix expression after each iteration of the outermost for loop. For each step circle the token in the infix experession that is being processed.

Infix	Stack	Postfix
5 * (6 + 4) / (8 + 2) - 7		
5 * (6 + 4) / (8 + 2) - 7		
5 * (6 + 4) / (8 + 2) - 7		
5 * (6 + 4) / (8 + 2) - 7		
5 * (6 + 4) / (8 + 2) - 7		
5 * (6 + 4) / (8 + 2) - 7		
5 * (6 + 4) / (8 + 2) - 7		
5 * (6 + 4) / (8 + 2) - 7		
5 * (6 + 4) / (8 + 2) - 7		
5 * (6 + 4) / (8 + 2) - 7		
5 * (6 + 4) / (8 + 2) - 7		
5 * (6 + 4) / (8 + 2) - 7		
5 * (6 + 4) / (8 + 2) - 7		
5 * (6 + 4) / (8 + 2) - 7		
5 * (6 + 4) / (8 + 2) - 7		
5 * (6 + 4) / (8 + 2) - 7		

Joanna Klukowska
joannakl@cs.nyu.edu

Problem 4 (? points) .

Write a code fragment that list all possible subsets of a set 0,1,2,...,31 (you can just print each subset to the standard output stream).

Use either Java or C++. Do no use the `BitSet` or `bitset` classes in those languages.

Joanna Klukowska
joannakl@cs.nyu.edu

Problem 5 (? points).

A basic max-heap does only supports additions and removal of the max element. Explain how we can implement the `update` operation that, given an index replaces the value stored in that index with a new value. The heap should be valid after that operation completes. The `update` should perform in $O(\log N)$.

Joanna Klukowska
joannakl@cs.nyu.edu

Problem 6 (? points).

The following implementation of the disjoint set data structure uses two arrays:

one, called `p`, to store the connectivity information about the sets,

the other, called `rank`, to store approximate height of the tree representing the set.

Add code to this implementation that would provide implementation for two functions called `getNumberOfSets()` and `setSize()`. Each function should be $O(1)$. You can use additional storage, if you wish. Indicate only the changes that you need to make to the functions below.

If you prefer to code in C, use the code below. If you prefer Java, then use the code on the next page. Make it clear which code you are working with.

```
class UnionFind {
private:
    vector<int> p, rank;

public:
    UnionFind(int N) {
        rank.assign(N, 0);
        p.assign(N, 0);
        for (int i = 0; i < N; i++)
            p[i] = i;
    }
    int findSet(int i) {
        return (p[i] == i) ? i : (p[i] = findSet(p[i]));
    }
    bool isSameSet(int i, int j) {
        return findSet(i) == findSet(j);
    }
    void unionSet(int i, int j) {
        if (!isSameSet(i, j)) {
            int x = findSet(i), y = findSet(j);
            // rank is used to keep the tree short
            if (rank[x] > rank[y]) {
                p[y] = x;
            }
            else  {
                p[x] = y;
                if (rank[x] == rank[y])
                    rank[y]++;
            }
        }
    }
}
```

Joanna Klukowska
joannakl@cs.nyu.edu

```
1
2  class UnionFind {
3      private Vector<Integer> p, rank;
4
5      public UnionFind(int N) {
6          p = new Vector<Integer>(N);
7          rank = new Vector<Integer>(N);
8          for (int i = 0; i < N; i++) {
9              p.add(i);
10             rank.add(0);
11         }
12     }
13
14     public int findSet(int i) {
15         if (p.get(i) == i) return i;
16         else {
17             int ret = findSet(p.get(i));
18             p.set(i, ret);
19             return ret;
20         }
21     }
22
23     public Boolean isSameSet(int i, int j) {
24         return findSet(i) == findSet(j);
25     }
26
27     public void unionSet(int i, int j) {
28         if (!isSameSet(i, j)) {
29             int x = findSet(i), y = findSet(j);
30             // rank is used to keep the tree short
31             if (rank.get(x) > rank.get(y)) {
32                 p.set(y, x);
33             }
34             else                            {
35                 p.set(x, y);
36                 if (rank.get(x) == rank.get(y)) rank.set(y, rank.get(y) + 1);
37             }
38         }
39     }
40 }
41
```

Joanna Klukowska
joannakl@cs.nyu.edu

Problem 7 (? points).

Write an algorithm for the following problem:

The $N \times M$ grid of integers gives terrain elevation. Given a water level L, every cell with the height (=elevation) $\leq L$ is below the water. The islands are the cells above the water.

Determine the number of islands.

Joanna Klukowska
joannakl@cs.nyu.edu

SCRAP PAPER NAME _____

Joanna Klukowska
joannakl@cs.nyu.edu

SCRAP PAPER NAME _____