Name: _____

The exam consists of six problems and *one extra credit question*; you only need to solve five problems. You *must* do problems 1, 2, and 3; the remaining two may be chosen from problems 4 – 6. Please indicate the problems you want to have graded by circling the problem numbers — otherwise, I will just grade the first five problems you worked on.

The following reference materials are provided on the last page of the exam: the statement of the Master Theorem and some summation formulas.

- You **have** 120 minutes.
- You **may use only** a calculator, pencil, or pen; you may not use a phone as a calculator.
- You **must** show all calculations. If you fail to show your work, you will receive no credit.
- You **must** put away all notes and papers and close all bags.

1. **(REQUIRED)** Consider the following algorithm for binary search on a sorted array A of length n. It returns TRUE if x is in A and FALSE otherwise.

```
BINARY-SEARCH(A, x)
 1  n = A.length
 2  if n == 1
 3       if A[n] == x
 4            return TRUE
 5       else
 6            return FALSE
 7  else
 8       q = ⌊n/2⌋
 9       if x ≤ A[q]
10            return BINARY-SEARCH(A[1..q], x)
11       else
12            return BINARY-SEARCH(A[q+1..n], x)
```

(a) Derive a recursion for the running-time of BINARY-SEARCH.

(b) Use a recursion tree to 'guess' an asymptotic bound for the recursion.

(c) Use the substitution method (proof by induction) to prove your guess is correct.

2. **(REQUIRED)** The MERGE-SORT algorithm recursively sorts the left and right halves of an n-long array A and calls the MERGE function to combine the two sorted halves into a single, sorted, n-long array. A simplified version of the main loop of MERGE is given below:

```
1  i = 1
2  j = 1
3  for k = 1 to n
4       if L[i] ≤ R[j]
5            A[k] = L[i]
6            i = i + 1
7       else A[k] = R[j]
8            j = j + 1
```

Assume L and R are $n/2$-long, sorted arrays; then when the loop terminates, A should contain the elements of L and R in sorted order. Use a loop invariant to prove the correctness of the loop.

3. **(REQUIRED)** Several of your friends have decided to hike the Appalachian Trail next summer. They want to hike as much of the trail as possible per day, but do not want to hike in the dark. On a map, they've identified a large set of good spots for camping, and are considering the following system for deciding when to stop each day. Each time they come to a potential camp site, they determine whether they can make it to the next one before nightfall. If they can make it, then they keep hiking; otherwise, they stop. They believe this system is optimal in that it minimizes the number of camping stops they will have to make. Show that their algorithm has the greedy choice property.

You may consider the trail to be a line segment of length L, assume that your friends can hike d miles per day irrespective of terrain and weather, and that the potential campsites are located at distances $x_1, x_2, \ldots, x_n$ miles from the start of the trail. In addition, you may assume that the campsites are never more than d miles apart, that the first campsite is d or fewer miles from the start of the trail, and the n^{th} campsite is d or fewer miles from the end of the trail.

4. The algorithm P-SUM computes the sum of the elements of an array L of length n:

```
P-SUM(L)
1  n = L.length
2  if n == 1
3        return L[1]
4  c = ⌊n/2⌋
5  x = spawn P-SUM(L[1 .. c])
6  y = P-SUM(L[c + 1 .. n])
7  sync
8  return x + y
```

(a) Determine the work $T_1(n)$ for P-SUM by deriving and solving a recurrence.

(b) Determine the span $T_\infty(n)$ by deriving and solving a recurrence.

(c) What is the parallelism of P-SUM?

5. Two friends in CMSC 201 decided to implement Dijkstra's algorithm, but, unfortunately, didn't understand the importance of using a min-heap data structure. Why not just use an array? They came-up with the following scheme. Let $n = |V|$; create an n-long array A of pointers, and initialize $A[i]$ to point to the i^{th} node of the graph, $i = 1, 2, \ldots, n$. It's easy to implement REDUCE-KEY(x, d'): assuming $d' < x.d$, simply set $x.d \leftarrow d'$, and RELAX is still $O(1)$. To implement EXTRACT-MIN, scan the entire array to find the entry $A[j]$ that points to the node x with the smallest $x.d$ value; return x and set $A[j]$ to NIL.

(a) What is the running time of Dijkstra's algorithm using this simple data structure?

(b) How does the running time using the simple data structure compare to using a min-heap?

6. A *hamiltonian cycle* in an undirected graph $G = (V, E)$ is a simple cycle that contains each vertex in V. A graph that contains a hamiltonian cycle is said to be *hamiltonian*. The decision problem HAM-CYCLE is: given $G = (V, E)$, is G hamiltonian? Show that HAM-CYCLE is in NP.

Extra Credit You are at your office holiday party when your supervisor asks, "What's your opinion on P = NP?" What do you say?

Theorem (Summation Formulas)**.**

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum_{i=0}^{n} r^i = \frac{1-r^{n+1}}{1-r}$$

Theorem (Master Theorem)**.** *Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence*

$$T(n) = aT(n/b) + f(n)$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

(a) *If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.*

(b) *If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.*

(c) *If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$.*