CPS 130 Final Exam

Spring 2001

9am-12pm, Saturday May 5
Closed book exam

NAME:____

Problem	Max	Obtained
1	15	
2 (a)	15	
3 (a)	5	
3 (b)	15	
4 (a)	15	
4 (b)	10	
5 (a)	5	
5 (b)	5	
5 (c)	15	
Total	100	

Comments:

- You can use any of the algorithms covered in class without describing them.
- When asked to describe an algorithm it is completely ok to do so with words (and a few accompanying pictures if it helps the description)—it is *not* recommended to write (pseudo-) code.

HONOR CODE

I have obeyed the honor code.

SIGNATURE:____

[*15 points*] **Problem 1:**

Show using induction (the substitution method) that the recurrence

$$T(n) = \begin{cases} 2 \cdot T(n/2) + n \log n & \text{if } n > 2 \\ 1 & \text{otherwise} \end{cases}$$

has solution $T(n) = O(n \log^2 n)$.

[*15 points*] **Problem 2:**

We want to maintain a data structure $\mathcal{D}$ representing an infinite array of integers under the following operations:

- $\textsc{Init}(\mathcal{D})$: Create a data structure for an infinite array with all entries being zero.
- $\textsc{Lookup}(\mathcal{D}, x)$: Return the value of integer with index x.
- $\textsc{Update}(\mathcal{D}, x, k)$: Change the value of integer with index x to k.
- $\textsc{Max}(\mathcal{D})$: Return the maximal index for which the corresponding integer is non-zero.
- $\textsc{Sum}(\mathcal{D})$: Return the sum of all integers in the array.

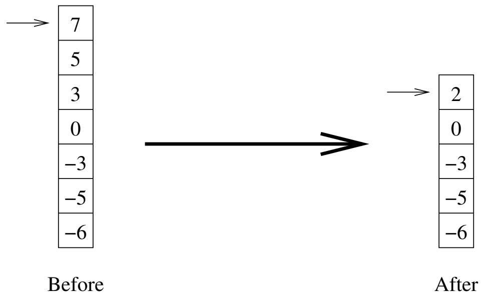
Describe an implementation of $\mathcal{D}$ such that Init, Max, and Sum runs in $O(1)$ time and Lookup and Update in $O(\log n)$ time, where n is the number of non-zero integers in the list.

[*20 points*] **Problem 3:**

An *ordered stack* $\mathcal{S}$ is a stack where the elements appear in increasing order. It supports the following operations:

- Init($\mathcal{S}$): Create an empty ordered stack.
- Pop($\mathcal{S}$): Delete and return the top element from the ordered stack.
- Push($\mathcal{S}$, x): Insert x at top of the ordered stack and reestablish the increasing order by repeatedly removing the element immediately below x until x is the largest element on the stack.
- Destroy($\mathcal{S}$): Delete all elements on the ordered stack.

The following shows an example of an ordered stack and the same stack after performing a Push($\mathcal{S}$,2) operation (the order is reestablished by removing 7, 5, and 3)



Like a normal stack we implement an ordered stack as a double linked list (maintaining a pointer to the top element).

a) What is the worst-case running time of each of the operations Init, Pop, Push, and Destroy?

b) Argue that the amortized running time of all operations is $O(1)$.

[*25 points*] **Problem 4:**

A *palindrome* is a string that reads the same from front and back. Any string can be viewed as a sequence of palindromes if we allow a palindrome to consist of one letter.

Example: "bobseesanna" can e.g be viewed as being made up of palindromes in the following ways: "bobseesanna" = "bob" + "sees" + "anna" "bobseesanna" = "bob" + "s" + "ee" + "s" + "anna" "bobseesanna" = "b" + "o" + "b" + "sees" + "a" + "n" + "n" + "a"

We are interested in computing $MinPal(s)$ defined as the minimum number of palindromes from which one can construct s (that is, the minimum k such that s can be written as $w_1 w_2 \dots w_k$ where $w_1, w_2, \dots, w_k$ are all palindromes).

Example: $MinPal$("bobseesanna")=3 since "bobseesanna" = "bob" + "sees" + "anna" and we cannot write "bobseesanna" with less than 3 palindromes.

We can compute $MinPal(s)$ using the following formula

$$MinPal(s[i,j]) = \begin{cases} 1 & \text{if } s[i,j] \text{ is palindrome} \\ \min_{i \le k < j}\{MinPal(s[i,k]) + MinPal(s[k+1,j])\} & \text{otherwise} \end{cases}$$

which can be implemented as follows

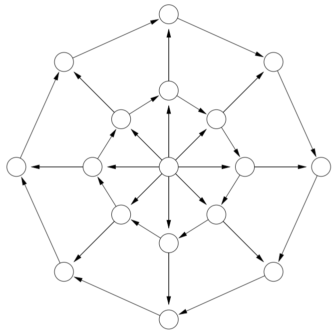
```
MinPal(i,j)

  b=i, e=j
  WHILE b<e and s[b]=s[e] DO
    b=b+1
    e=e-1
  OD
  IF b>=e THEN RETURN 1

  /* s[i,j] is not palindrome */

  min=j-i+1
  FOR k=i to j-1 DO
    r=MinPal(i,k)+MinPal(k+1,j)
    IF r<min THEN min=r
  END
  RETURN min

END
```

a) Show that the running time of MINPAL(s) is exponential in the length n of s.

b) Describe an $O(n^3)$ algorithm for solving the problem.

[*25 points*] **Problem 5:**

A *wheel-graph* is a directed graph of the following form:



More precisely, a wheel-graph consists of a center vertex c with k outgoing "spokes" of s outward oriented edges each. Furthermore, the ith vertex ($i = 2, 3, \ldots, s+1$) of all the spokes are connected to form a directed cycle. All cycles are oriented the same way (refer to the figure, in which $k = 8$ and $s = 2$).

a) What is the number of edges m in a wheel-graph as a function of the number of vertices n?

Assume we are given a wheel-graph with positive integer edge-weights. We want to find the length of the shortest paths from the center c to all other vertices.

b) How long time would Dijkstra's algorithm use to solve the problem (as a function of n)?

c) Describe a more efficient algorithm for solving the problem. Remember to argue for both running time and correctness.