CSE 326 – Data Structures
Winter 2004. Dry assignment #4
Due date: 2/27/04

1. Given the following initial array of integers, show the contents of the array after each of the operations below. Assume that the operations are performed one after the other, that is, the output of operation i is the input to operation i+1.

-∞	4	10	3	9	2	8	1	11	5	7

Note: While it is enough to show the content of the array after each operation, it is recommended to list some intermediate states too – this way you can still get partial credit if you make a mistake along the way. If you provide intermediate states, be sure to clearly mark the final state after each operation.

I. BuildHeap( );

II. DeleteMin( );

III. IncreaseKey (5, 7) [5 is the index of the element whose key should be increased by 7. Indices are counted from 0 - where the -∞ is located];

IV. Insert (1);

V. Insert (6);

2. a. Prove that a binomial tree B_k has binomial trees B_0, B_1, …, B_{k-1} as children of the root.

b. Draw a binomial queue that includes the first 13 even numbers 2, 4, 6, …, 26, and a binomial queue that includes the first 7 odd numbers 1, 3, 5, …, 13. (There are many ways to do it – you need to show one way.) Then, merge the two binomial queues and draw the resulting binomial queue.

3. Given an array of *n* integers, and an additional integer *s*, your goal is to determine if there is a pair of two elements in the array, *a[i]* and *a[j]*, such that $i \neq j$ and $a[i]+a[j]=s$.

For example, if *A= {4, 8, -3, 2, 15, 7}* and *s=9* then the output should be *'yes'* (since 2+7=9). If *A= {4, 8, -3, 2, 15, 7}* and *s=8* then the output should be *'no'* (since there is no pair of two different elements in the array whose sum is *8*).

Rules: 1. You are not allowed to change the array in order to answer.
2. If the answer is *'yes'* then there is no need to report *a[i]* and *a[j]* nor the indices *i* and *j*, only to determine the positive answer.

For each of the following complexity constraints, describe your algorithm (in words or pseudo-code, whatever you prefer), and justify its time and space complexity. You can assume that the elements are indexed *1* to *n*, i.e., *a[1]* is the leftmost element and *a[n]* is the rightmost element.

a. Give an algorithm that uses $O(1)$ space (in addition to the array) and finds an answer in time $O(n^2)$.
b. Give an algorithm, based on *hashing* that uses $O(n)$ space and whose time complexity is $O(n)$ [assuming that each insert/find hashing operation takes $O(1)$ time].

4. A set includes the numbers 9, 21, 8, 25, 26, 12, 13. These numbers are inserted into a hash-table of size 13, using the hash function h(x)=x mod 13. No deletions were performed. Collisions are resolved using linear probing, that is, $h_i(x)=h(x)+i$. The insertion order of the numbers into the hash-table is not known (i.e., it is not known which number was inserted first, which one was inserted second, etc.). What is known is that after all the numbers have been inserted, the contents of the table are:

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
content	**13**	**26**	**25**						**8**	**21**	**9**		**12**

For each of the claims 4.1-4.4, determine if it:

(i) holds for any insertion order that produces the above table, or if

(ii) there is no insertion order that produces the above table for which the claim holds, or if

(iii) there exists an insertion order that produces the above table for which the claim holds and another insertion order that produces the above table for which the claim does not hold.

Justify your answers!

4.1 The number 25 was inserted last.

4.2 At least three numbers were inserted before 25.

4.3 8 was inserted before 12.

4.4 9 was inserted before 8.

5. A *directed* graph is given by the following adjacency matrix, M, in which M(i,j)=1 if and only if there is an edge from i to j. Find a topological sort of the graph.

	A	B	C	D	E	F
A	0	1	0	0	0	0
B	0	0	1	0	0	0
C	0	0	0	1	0	0
D	0	0	0	0	0	0
E	0	1	1	0	0	0
F	1	0	1	0	0	0