08.71.23/24 Tölvunarfræði 2/2a
(English exam)

Makeup exam **August 15th, 2002**

Professor: Hjálmtýr Hafsteinsson **Time: 13^{30} ? 16^{30}**

The first 5 problems are for all students (both from Tölvunarfræði 2 and 2a) Problem 6 is only for students in Tölvunarfræði 2, but problem 7 is only for students in Tölvunarfræði 2a (engineering students). In both cases ***five best problems out of six count****. All problems have the same value.*

All written materials and a calculator are allowed.

- Note that when asked to "Describe" or "Show" then it is enough to do that in words and with drawings. If you are to write C++ code you will be asked for that specifically.
- Give supporting arguments for all answers and remember that it is not necessary to write up definitions from the book.

1. In Union-find we use path compression to shorten the path from node *p* to the root of the tree that it is in. In full path compression we make all the nodes on the path from *p* to the root point to the root. This requires two passes, first one to find the root, and then another to make all the nodes point to it. Let us consider a new method, *path reversal.* In it we make all the nodes on the path from *p* to the root (including the root!) point to *p*, not the original root. This will make *p* into a new root of the tree.

a) Show, with a drawing, an example of this new method, where *p* is originally 4 nodes away from the root.

b) Show an implementation of this method (similar to Program 1.4 on page 19 in the textbook).

c) Describe a bad case for this method.

2. You are given a vector containing *n* items, where you do not know the value of *n*. The items, that are positive integers, are in accending order. The vector itself is very large and the items only occupy the first *n* locations of it. The following locations all contain the value -1. Below is a small example of such a vector, where the number of items turns out to be 5:

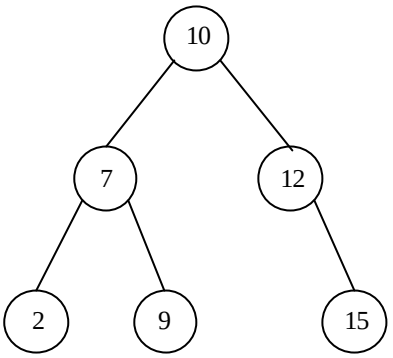
2	4	5	5	6	-1	-1	-1	-1	...

Write a function in C++ that gets as parameter a pointer to the vector and looks for the value *x* in the vector in time O(log *n*). Note that the function does not know the value of *n*.

3. Write a function in C++ that gets as input a pointer to a circular singly-linked list and two values *x* and *v*. The function searches for the value *x* in the list and puts the value *v* immediately behind it in the list (in a new node). If *x* was found then the functions returns a pointer to the new node, otherwise it returns 0 (or *NULL*). Remember to take into account all special cases (such as empty list, etc.).

4. Decribe a method to find the kth smallest item in a n item unsorted vector in O(n*log k) time. Use a heap of limited size.

5. You are given the following binary search tree:



a) Show at least 4 different series of insertions of the values in the tree, that would result in this binary search tree.

b) How many different insertions are there that would result in this binary search tree?

Only for students in Tölvunarfræði 2:

6. Below is a slightly different implementation of the Quicksort sorting method:

```
void quicksort( Item a[], int l, int r)
{   while( l < r ) {
        int i = partition( a, l, r);
        quicksort( a, l, i );
        l = i+1;
    }
}
```

a) Show that this version sorts correctly.

b) Recursive functions store the parameters on the system stack and therefore often use more memory than nonrecursive ones. Describe an instance where the above implementation uses O(n) space on the system stack.

c) Change the code above so that we only need O(log n) space on the system stack in the worst case.

Only for students in Tölvunarfræði 2a (engineering students):

7. On page 171 in the textbook there is an array implementation of a queue.

a) In the implementation in the book there is no check if the operation is legal (e.g. remove an item from an empty queue or put an item in a full one). Add these checks, using `if`-statements and write out a message if there is an error. Justify the conditions you use in the `if`-statements.

b) Add the operation `pop`, that removes the item that was last entered (using `put`). Implement the operation with error checks as described in part *a)*.