01OGD Algorithms and Programming

16/06/2015 – Part I: Theory (12 points)

1. (1+1 points)

Sort in ascending order the following array of integers:

12 91 2 41 13 3 39 43 29 17 71 10 18 6 8 100

- resorting to selection-sort
- resorting to merge-sort.

Report relevant steps.

2. (2 points)

Suppose to have an initially empty priority queue implemented with a heap. Given the following sequence of integers and * character: 12 23 29 61 * * 18 * 34 9 13 * * 14 * 21 27 28 where each integer corresponds to an insertion into the priority queue and character * corresponds to an extraction of the minimum, show the priority queue after each operation and return the sequence of values extracted.

3. (2 points)

Using a greedy approach find an optimal Huffman code for the following characters. Each character has an associated integer frequency:

a: 26 b: 6 c: 4 d: 3 e: 22 f: 13 g: 14 h: 9 i: 18 l: 10

4. (1.5 points)

Write the prefix, postfix and infix (without brackets) form of the following arithmetic expression. Visit the corresponding binary tree.

((A + B) – (C – D)) * E) /((F * G) / H – I)

5. (1 point)

Suppose that all integers in the range 1-500 are stored in a Binary Search Tree. Suppose you are looking for key 231. Which ones among these sequences **cannot** be a sequence found during search? Why?

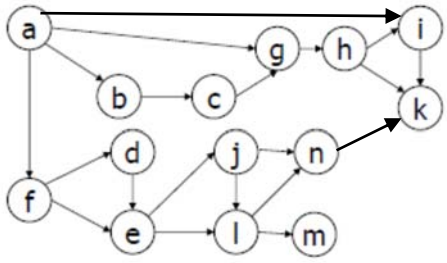
250 200 240 260 255 220 230 231

450 100 400 150 350 200 250 231

270 100 200 300 400 450 260 350 220 225 230 231

6.

Suppose to have the following DAG: considering node **a** as starting node



- visit it in breadth-first (**1.5 points**)
- sort it in reverse topological order **(2 points).**

Whenever necessary consider nodes in alphabetical order. Assume a topological order for the adjacency list.

Algorithms and Programming

Examination Test

Programming Part
16 June 2015

No books or notes are allowed. Examination time: 100 minutes.

The programming part includes two *possibilities*:

- **Exercise number** 1 **(traditional) of a maximum value equal to** 18 **points.**
- **Exercises** 2**,** 3 **and** 4 **(easier) with a total maximum value of** 12 **points.**

Intermediate solutions are forbidden.

1 (18.0 points)
Write a C application able to manage a ski resort with the following specifications.

Each skier has a key-card whose identifier (the `cardId`) is an integer number. Each chairlift (or cable-car) has a string as identifier (the `chairliftId`) made up of 10 alphabetic characters. Each chairlift has a car-reader to enable a skier to use the chairlift.

The system receives from standard input (a keyboard) data from the chairlift car-readers, with the following format:

```
chairliftId cardId time
```

and it has to grant the authorization to use the chairlift. Time is introduced as an integer value which indicates the number of minutes passed from the midnight (`00:00` hour) of the same day. The target is to avoid cheaters, i.e., different skiers using chairlifts with the same key-card. As a result the authorization is given only if the key-card has not been used on the same chairlift for a specific time interval. Time intervals are specific for each chairlift and vary depending on the position and length of the chairlift itself.

Time intervals have to be read from file. The file name is received by the application on the command line. It has the following format:

```
chairliftId   timeInterval
```

where `timeInterval` is an integer value in minutes.

After the initial set-up, the application has to:

- Deliver a function such as

  ```
  int authorize (int cardId, char *chairliftId, int time);
  ```

  that allows the skier with card `cardId` to use the chairlift `chairliftId` or force him/her to wait whenever the time delay for that chairlift has not been respected.
- Maintain, into main memory, the list of all skiers that have used each chairlift and, for each chairlift, the number of times a skier has used the chairlift.
- Maintain, into main memory, the list of all chairlifts used by each skier and, for each chairlift the time of last authorization.

Notice that:

- As the number of skiers is quite high and it increases during the day, all operations have to be performed with asymptotic costs at most logarithmic in the worst case.
- As the number of chairlifts is specified by the file storing all chairlift delays and it is somehow limited, there are no efficiency constraints in terms of the number of chairlifts.

continue

2 (4.0 points)
Write the function

```
void mul (int *v1, int *v2, int n, int **pv);
```

which multiplies the decimal number stored in the array `v1` by the decimal number stored in `v2`. Numbers are stored in the arrays one digit for each element. n is the number of digits of the two numbers. The function must apply the standard sum-and-shift algorithm, taking care of carry values, as indicated by the following example:

```
      0 3 2 x
      2 4 3 =
------------
0 0 0 0 9 6 +
0 0 1 2 8   =
0 0 6 4
-----------
0 0 7 7 7 6
```

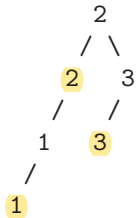
Note that, to avoid overflow, the final product has to be represented on $2 \cdot n$ digits. Function `mult` has to allocate the array referenced by the pointer `pv`, and it has to return the result in this array.

3 (4.0 points)
Write function

```
void doubleTree (struct node *root);
```

that for each node in a binary tree, creates a new duplicate node, and inserts the duplicate as the left child of the original node. For example, the tree

is changed to the following one:



Write the node data structure `struct node` storing integer keys.

4 (4.0 points)
A set of products-and-producers is stored in a list of lists.
The main list includes the producers name (a string of 30 characters at most). The secondary lists store, for each producer, the product name (a string of 30 characters at most) and its price (a real value). The main list and all secondary lists are alphabetically ordered (by producer and product names, respectively). All producer and product names are unique in the entire data base.
Write a single recursive function in C which receives a producer name, a product name, and any other parameter the candidate may want to add. This function has to print-out the price of the product of the specified producer if it exist, and an errore message it if does not exist.