01OGD Algorithms and Programming

02/09/2015 – Part I: Theory (12 points)

1. (**1 point**)
Sort with insertion sort in ascending order the following array of integers. Report relevant steps.

22 11 12 51 23 3 19 43 29 17 81 100 18 60 8 10.

2. (**1 point**)
Given the following set of activities, where the *i*-th activity is identified by the pair start time (s_i) and end time (f_i):

i	s_i	f_i	***i***	s_i	f_i
1	0	6	5	1	9
2	7	10	6	11	15
3	2	4	7	3	7
4	5	9	8	4	10

Find by means of a greedy algorithm the maximum set of mutually compatible activities.

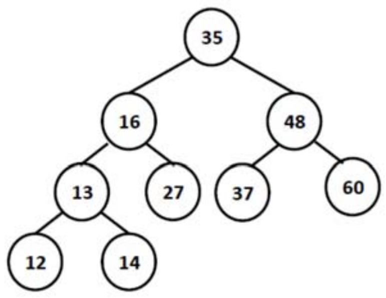
3. (2 points)
A binary tree has 15 nodes. Its visits return the following sequences of keys:

preorder	30	18	33	12	16	17	19	7	21	10	11	9	3	13	15
Inorder	16	12	17	33	7	19	18	21	30	11	3	9	13	10	15
postorder	16	17	12	7	19	33	21	18	3	13	9	11	15	10	30

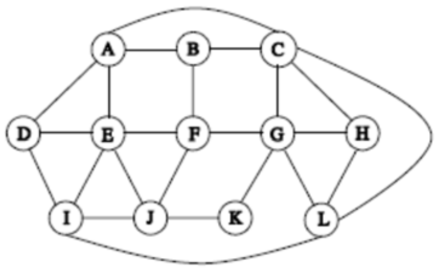
Draw the binary tree.

4. (2 points)
Given the sequence of keys I B U X E G Z H Y, where each character is identified by its index in the English alphabet (A=1, ..., Z=26), draw the final configuration of an initially empty hash table of size 19 where insertion of the previous sequence occurs character by character. Assume open addressing with double hashing. Define suitable hash functions. Show relevant intermediate steps.

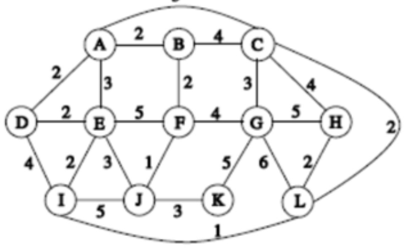
5. (2 points)
Partition the following BST around its median key:



6. (2 points)
Visit in depth-first the following undirected graph. Start from node **A.** Label nodes with discovery and end-processing times in the format time1/time2. If necessary, assume an alphabetical order for the nodes.



7. (2 points)
Using Prim's algorithm starting from node A, find a minimum-spanning tree for the following undirected and weighted graph. Show relevant intermediate steps.



Algorithms and Programming

Examination Test

Programming Part
02 September 2015

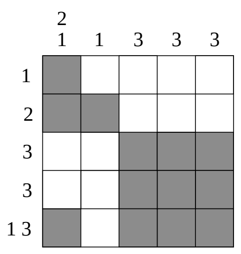
No books or notes are allowed. Examination time: 100 minutes.

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)
Nonograms, also known as Hanjie or Griddlers, are picture logic puzzles in which cells in a grid must be colored or left blank according to numbers at the side of the grid to reveal a hidden picture.
For example, a clue of "`4 8 3`" would mean there are sets of four, eight, and three filled squares, in that order, with at least one blank square between successive groups. To solve a puzzle, one needs to determine which cells will be black and which will be empty.
An example (already solved) is reported on the left-hand side of the following figure.



```
5
1  1
1  2
1  3
1  3
2  1  3
5
2  2  1
1  1
1  3
1  3
1  3
```

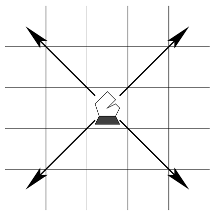
```
5  5
1  0  0  0  0
1  1  0  0  0
0  0  1  1  1
0  0  1  1  1
1  0  1  1  1
```

Nonograms are stored on files with the following format. The first line reports the number of row of the puzzle. For each row of the puzzle, is then stored in the file a single line containing integer numbers. The first number specifies the number of groups of black cells on that row of the puzzle. All following numbers specifies the length of each set of black cells. After that, the same information is stored for the columns of the puzzle.
For example, the puzzle represented on the left-hand side of the previous figure can be specified with the data file reported at the center of the figure. The line "`2 1 3`" indicates that in the corresponding row of the puzzle (the fifth) there are 2 blocks of black cells, of size 1 and 3, respectively.

Write a C program able to solve the nonogram puzzle using a recursive procedure. The program receives two parameters on the command line. The first one is the input file name storing the requirements to satisfy while building the final picture (whose format is the one reported at the center of the previous figure). The second one is the output file name in which the final puzzle has to be stored using `0` and `1` to indicate empty and black cells, respectively. The first line of the output file indicates the size of the puzzle. An example of such an output file is reported on the right-hand side of the previous figure.

continue

2 (2.0 points)
In the chess game a bishop can move any number of squares diagonally as represented by the following figure (left-hand side). Moreover, in the chess game each piece (pawn, knight, bishop, rook, queen and king) has a specific value (e.g., 1



	0	1	2	3
0	0	3	4	0
1	1	0	6	6
2	1	3	9	0
3	0	0	3	1

for a pawn, 3 for a knight, etc.). Empty cells are assigned a zero value.
Write an iterative C function that given a matrix containing integer values, is able to print-out (on standard output) the coordinates of the empty cell in which a bishop can be placed such that the sum of the values of all pieces placed on the same diagonal and inverse diagonal is maximum.
For example, given the matrix at the right-hand side of the previous figure, the bishop should be positioned on the element $[3, 1]$ (with a sum equal to 16).

3 (4.0 points)
Write function

```
int treeIsomorph (struct node *t1, struct node *t2);
```

that given two binary trees, referenced by `t1` and `t2`, it returns a true value if they are structurally identical, i.e., they are made of nodes with the same values arranged in the same way.
Suppose the tree has dynamic strings as keys. Write down the data structure of the C node.

4 (6.0 points)
A map is represented as an matrix of integer values. The starting cell of the map is always at the top-left corner, while the ending cell is always at the bottom-right corner. It is possible to walk the map by moving from one cell to any adjacent element, but each cell has to be visited only once.
Write a recursive C function that, receiving the matrix as a parameter (and any other parameter the candidate may consider as necessary), it is able to find a path from the starting point to the ending point whose sum of the visited cell is maximum. If there are more paths with the same sum, the function has to retain the shortest path.
An example, of the matrix (right-hand side) and the solution (left-hand side) is reported in the following figure.

1	2	-3
9	-9	7
0	1	4