Coursework 3. Submission deadline 10/01/19

1 Merging rows of a matrix (20 points)

Write a program whose input is an $n \times n$ matrix A with each of its rows being sorted. The program should merge all the rows into a one-dimensional sorted array.

For example if A is

```
1,3,5
1,2,6
4,7,8
```

Then the result of the merging is list $[1, 1, 2, 3, 4, 5, 6, 7, 8]$.

The runtime of the program must be $O(n^2)$.

2 Processing elements of a binary tree (20 points)

Write a program whose input is a binary tree T (not necessarily a binary search tree) such that the content of each node of T is an integer number. The program should print the sum of contents that are even.

Hint: This is a minor modification of the algorithm computing the sum of elements of the tree algorithms (considered in class).

3 Binary search tree approximation(20 points)

In this task we will consider a binary search tree T. One algorithm considered in class takes as input T and a number x and checks whether T has a node whose content is x.

In this section, we consider a task of finding a node of T whose content y is *closest* to x. The 'closest' means that $abs(x - y)$ is smallest possible among all the nodes of T (abs is the absolute value of a number). For example, if the numbers stored in the nodes of T are $1, 2, 5, 8$ and $x = 3$ then the closes value to x is 2.

Write a program whose input is a binary search tree T and an integer number x. The program should print the content y of a node of T that is closest to x. The runtime of the program must be $O(d)$ where d is the depth of the tree.

4 Recognizing a star (20 points)

A graph G is a star if it has a vertex x adjacent to every other vertex and x is *the only* neighbour of every other vertex.

Given the adjacency matrix of a graph G, design an $O(n^2)$ algorithms that prints YES if G is a star and NO otherwise.

5 Processing the list of edges

5.1 Listing all the edges (5 points)

Write an $O(n^2)$ program whose input is the adjacency matrix of a graph G. The program should print the list of all the edges (without repetition). For example, if the vertices of the graph are $0, 1, 2, 3, 4, 5$ with 0 adjacent to 1 and 5, 1 adjacent to 5 2 adjacent to 3 and 4 and 3 adjacent to 4 then the printed list should be something like.

```
0 1
0 5
1 5
2 3
2 4
3 4
```

5.2 Removal of an edge (5 points)

Let G be a graph and e be an edge of G. We denote by $G \setminus e$ the graph obtained from G by removal fo e. That is, the vertices of $G \setminus e$ are the same as in G and the edges of $G \setminus e$ are all the edges of G but e.

Write a program whose input is the adjacency matrix of a graph G and two vertices i and j such that that are adjacent in G. The program should print the adjacency matrix of $G \setminus \{i, j\}$ (that is G with the edge between i and j being removed).

5.3 Listing all the bridges (10 points)

Let G be a *connected* graph and e be an edge of G. We say that e is a *bridge* of G if $G \setminus e$ is not connected (put it differently, the removal of e breaks the connectivity of G).

Assume that we have a function *Connect* whose input is the adjacency matrix of a graph and the output is *true* if the graph is connected and $false$ otherwise. Using this procedure, design an algorithm that prints all the bridges of G.

Hint: Use the algorithm for the first part of that question for exploration of all the edges. Rather than printing the curretly considered edge straightaway, use the *Connect* procedure to check whether the given edges is a bridge (the whole non-triviality of this task is how to apply *Connect* in this context). Print only those edges for which *Connect* retrns *true*.