Reserved Cells

Ex. 1	
Ex. 2	
Ex. 3	
Ex. 4	
Ex. 5	
Ex. 6	
Tot.	

Algorithms and Programming

26 June 2019

Part I: Theory

Register Number ______________ Family Name ______________________ First Name ______________________

Course: ◯ 01*OGDLP* 10 credit ◯ 02*OGDLM* 12 credit

No books or notes are allowed. Solve exercises directly within the reserved space. Additional sheets are accepted only when strictly necessary. Available time: 60 minutes.

1. (2.0 **points)**
   Evaluate analytically the worst case time complexity of the following algorithm:

```
void f (int A[], int n) {
  int i, j, temp;

  for (i=0; i<n-1; i++) {
    for (j=0; j<n-i-1; j++) {
       if (A[j] > A[j+1]) {
         temp = A[j];
         A[j] = A[j+1];
         A[j+1] = temp;
       }
    }
  }
  return;
}
```

2. **(**2.0 **points)**
   Given the following sequence of pairs, where the relation *i*-*j* means that node *i* is adjacent to node *j*:

   2-7 5-3 1-7 8-9 0-1 6-4 8-3 1-2 5-9 2-6 4-0 3-8

   apply an on-line connectivity algorithm with quick-find showing at each step the content of the array and the corresponding tree representation. Node names are integers in the range from 0 to 9.

   Describe the differences (in terms of logic and complexity) among quick-find, quick-union, and weighted quick-union.

3. **(1.5 point)**
A correct BST contains integer keys in the range 1-500. The user searches for key 271. Among these sequences, which are the ones that cannot occur? Why?

Sequence A :	14	212	391	372	313	295	279	211	237	265	275	271		
Sequence B :	373	352	315	297	283	191	237	265	279	278	269	271		
Sequence C :	73	152	189	227	499	391	237	365	259	348	266	303	277	271
Sequence D :	142	193	237	469	356	244	355	269	318	276	313	287	271	

4. **(1.5 points)**
   10 **credit course** (01$OGDLP$)
   Given an initially empty BST, perform the following sequence of operations on the BST leaves:

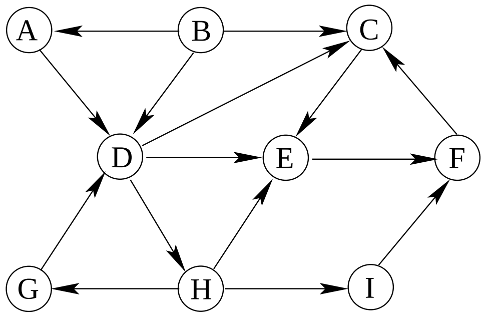
   $$+13 \quad +7 \quad +1 \quad +18 \quad +23 \quad +9 \quad +12 \quad +31 \quad +22 \quad +17 \quad +2 \quad +5 \quad -13 \quad -7 \quad -18$$

   where each + indicates a leaf insertion and each − a node extraction.

   12 **credit course** (02$OGDLM$)
   Using a greedy algorithm find an optimal Huffman code for the following symbols with the specified frequencies:

   $$A{:}15 \quad B{:}11 \quad C{:}13 \quad D{:}8 \quad E{:}7 \quad F{:}15 \quad G{:}9 \quad H{:}2 \quad I{:}17 \quad J{:}4$$

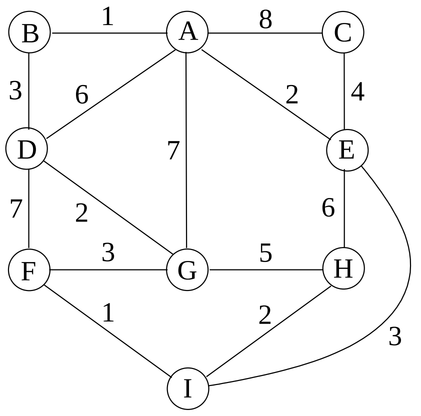
5. **(3.0 points)**
Given the following directed graph:



- **ONLY for the** 10 **credit course** (01*OGDLP*): Represent it as an adjacency list and as an adjacency matrix.
- **FOR EVERYBODY**: Visit it in breadth-first starting at node A.
- **FOR EVERYBODY**: Visit it in depth-first starting at node A. Label nodes with discovery and end-processing times in the format $time_1/time_2$. Redraw it labeling each edge as T (tree), B (back), F (forward), C (cross).
- **ONLY for the** 12 **credit course** (01*OGDLP*): Represent the reverse graph and find all strongly connected components.

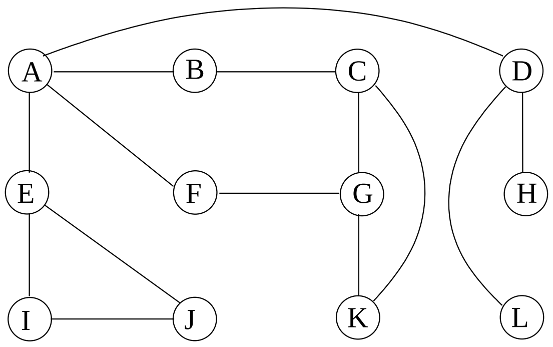
Whenever necessary consider nodes in alphabetical order.

6. **(2.0 points)**
10 **credit course** (01*OGDLP*)
Given the following undirected and weighted graph find a minimum spanning tree using Prim's algorithm starting from vertex *A*.



Draw the tree and return the minimum weight as a result. Show all relevant intermediate steps and all cuts generated during the process.

12 **credit course** (02*OGDLM*)
Given the following undirected and connected graph:



find all bridges and all articulation points.

If necessary, consider nodes in alphabetical order. Show all relevant intermediate steps.

Algorithms and Programming

26 June 2019

Part II: Program (12 point version)

At most one C manual is allowed. Available time: 120 minutes. Final program due by 11.00 a.m. of Monday the 1st of July; use the course portal page ("Elaborati" section) to upload it.

1 (2.0 **points)**
Write the function

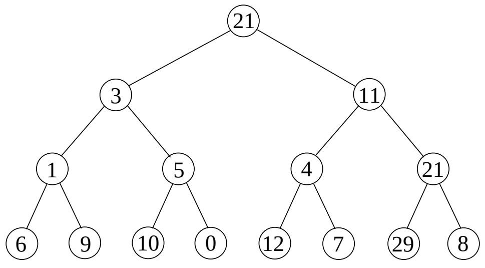
`void longestSubstring (char *in, char **out);`

which receives the string `in` and it returns the longest sub-string within `in` formed only by uppercase letters. The string `out` must be allocated by the function with the minimum correct length.
For example if
`in = "The longest ENGlISH word is:  PSEUDOpseudohypoparaTHYROIDISM"`
the function has to create and return
`out = "THYROIDISM"`.

2 (4.0 **points)**
Given a binary tree of integer keys, write the function

`void longestPath (node *root);`

which receives the pointer `root` to the tree root, and it prints the longest sequence of keys with strictly increasing values encountered by visiting the tree from root to leaves.
For example, given the following binary tree



the longest sequence of increasing keys include keys {3, 5, 10} or keys {11, 21, 29} at choice.
Report the C node structure to store the tree nodes.

3 (6.0 **points)**
Given an integer number `n`, write the function

`void nonIncreasing (int n);`

which prints all possible non-increasing sequences of positive integer numbers whose sum equals `n`.

For example if `n = 3` such sequences are: {1, 1, 1}, {2, 1}, and {3}.
If `n = 4` such sequences are: {1, 1, 1, 1}, {2, 1, 1}, {2, 2}, {3, 1}, and {4}.

Algorithms and Programming
26 June 2019
Part II: Program (18 point version)

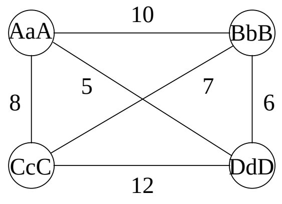
At most one C manual is allowed. Available time: 120 minutes. Final program due by 11.00 a.m. of Monday the 1st of July; use the course portal page ("Elaborati" section) to upload it.

Given `n` cities, we have to place `k` ATMs on different cities. To facilitate the citizens' activity, for each city we are interested in the distance between the city and the closest ATM. The program must place the `k` ATMs such that the maximum among all these minimum distances is minimized.

The city map is represented using an undirected unweighted graph, stored in a text file. The file reports the list of graph edges, one for each row. The names of the city are represented by the identifiers of the vertices, and are strings of unknown length, but limited to 100 characters. The distance (in kilometers) between cities are represented by the edge weights (positive integer values).

The following picture represents an input file and the corresponding graph, with four cities and the distances between them. Let us suppose we want to place 2 ATMs among these 4 cities so that the maximum amount all distances from a city to the closest ATM is minimized. The two ATMs should be placed in cities `CcC` and `DdD`. In such a situation the

```
AaA BbB 10
AaA CcC  8
AaA DdD  5
BbB CcC  7
BbB DdD  6
CcC DdD 12
```



minimum distance from `AaA` to the closest ATM is 5, from `BbB` is 6, and for `CcC` and `DdD` is 0. Thus the maximum distance from a city to the closest ATM is 6. All other possible placements of the 2 ATMs among the 4 cities lead to a larger distance.

The C program must:

- Receive on the command line the name of the file storing the graph and the number `k` of ATMs to place in the city map.
- Store the graph in a proper data structure.
- Solve the aforementioned problem, and print the best possible solution.