Reserved Cells

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

Algorithms and Programming

21 February 2020

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

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

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

   Show a possible implementation of quick-find, adopting a representation on an array of size `N`.

2. **(2.0 points)**
Given the following sequence of integers stored in an array:

16 13 14 11 12 9 10 7 8 5 6 3 4 1 2

sort it in ascending order with shell sort. Use the Knuth's sequence $h = 3 \cdot h + 1$ (1, 4, 13, etc.).

Show all relevant intermediate steps. List at least 2 sorting algorithms that have a worse and 2 algorithms that have a better asymptotic complexity? Motivate your answer (which are these asymptotic complexity?).

3. **(2.0 point)**
   10 credit course (01$OGDLP$)
   Convert the following expression from in-fix to pre-fix notation (Polish Notation) and from in-fix to post-fix notation (Reverse Polish Notation).

$$A * B - \{(C + D - E) * [((F - G) * H)/I]\}$$

   Write the two C functions that, given the binary tree used to represent the expression, can generate the pre-fix and the post-fix notations by visiting the tree.

12 **credit course** ($02OGDLM$)

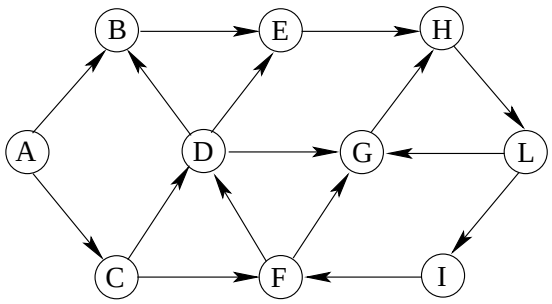
Given an initially empty Interval BST, insert in the leaves the following closed intervals:

$$[10,17] \quad [4,6] \quad [3,5] \quad [9,13] \quad [8,17] \quad [12,15] \quad [11,13] \quad [16,20] \quad [7,9] \quad [0,3] \quad [14,23]$$

After that, perform the following operations in sequence: Search an intersection with the interval $[19,21]$, and then with the interval $[1,2]$. Describe all recursion steps to perform these operations, and specify when it is necessary to recur on the right child and when to recur on the left one and why.

4. **(2.0 points)**
   Given the sequence of keys $CAPTAINMARVEL$, 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 tables of size 23 where insertion of the previous sequence occurs (character by character). Use open addressing with quadratic probing, with $c_1 = 1$ and $c_2 = 1$.

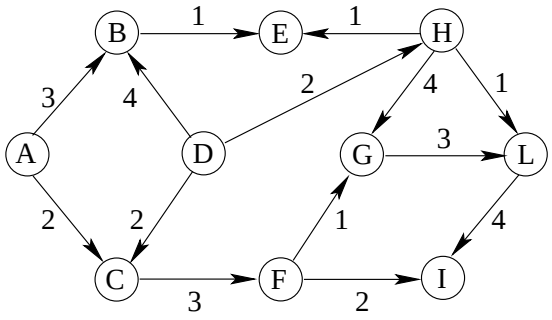
5. (2.0 **points**)
10 **credit course** ($01OGDLP$)
Given the following unweighted directed graph:



- Visit it in breadth-first starting from node A.
- 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), starting at node A.

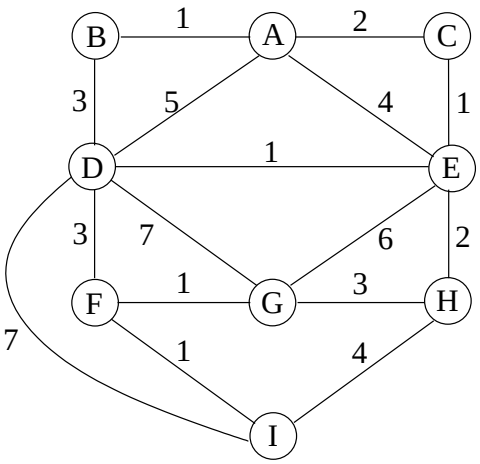
Whenever necessary consider nodes in alphabetical order.

12 **credit course** ($02OGDLM$)
Given the following weighted DAG:



Find the topological order of its vertices, and then all shortest paths from vertex A, using the simplified algorithm for directed acyclic graphs. If necessary, consider nodes in alphabetical order. Show all relevant intermediate steps.

6. (2.0 **points**)
Given the following undirected and weighted graph find a minimum spanning tree using **Prim** 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.

Define and explain what a "cut" and a "safe edge" are.

Algorithms and Programming
21 February 2020
Part II: Program (12 point version)

At most one C manual is allowed. Available time: 120 minutes. Final program due by 8.00 p.m. of Tuesday the 25th of February; use the course portal page ("Elaborati" section) to upload it.

1 (2.0 **points**)
Given a string `str` containing **only** lowercase letters, write the function

```
void histogram (char *str);
```

which prints all letters belonging to `str` along with the frequency of their occurrence in the string.

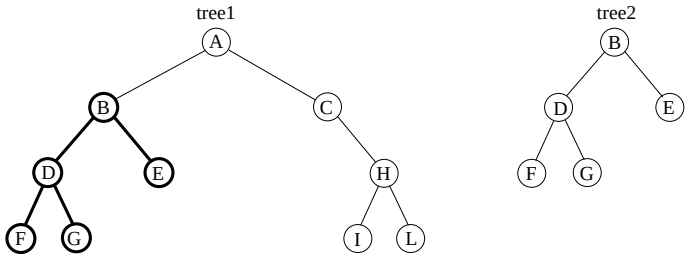
For example if
`str = "azaaabbnnccacczn"`,
the procedure must generate the following output
`"a":5`, `"b":2`, `"c":4`, `"n":3`, `"z":2`.

2 (4.0 **points**)
Given two binary trees rooted at `tree1` and `tree2`, write the function

```
int subtree (node_t *tree1, node_t *tree2);
```

which returns `1` if `tree2` is a subtree of `tree1`, and it returns `0` otherwise. Report the C node structure to store the tree nodes under the hypothesis that the key is an integer value.

For example, if `tree1` and `tree2` are the ones represented in the following figure, the function must return `1`, as `tree2` is a subtree of `tree1` (the one highlighted in bold).



3 (6.0 **points**)
Write the recursive function `generate` with the following prototype:

```
void generate (char *str, int n, int m);
```

The function receives a string `str` of decimal digits and two integers `n` and `m`. It prints all sets of digits belonging to `str`, which have **at most** `n` elements, and such that the sum of these elements is **exactly equal** to `m`.

Let us suppose that the function receives `str` = "1234567890", `n` = 2, `m` = 9. It must print the following sets: $\{9\}$, $\{9,0\}$, $\{0,9\}$, $\{1,8\}$, $\{8,1\}$, $\{2,7\}$, $\{7,2\}$, $\{3,6\}$, $\{6,3\}$, $\{4,5\}$, $\{5,4\}$.

Let us suppose that the function receives `str` = "111223", `n` = 2, `m` = 5. It must print the following sets: $\{2,3\}$, $\{3,2\}$, $\{2,3\}$, $\{3,2\}$.

Algorithms and Programming
21 February 2020
Part II: Program (18 point version)

At most one C manual is allowed. Available time: 120 minutes. Final program due by 8.00 p.m. of Tuesday the 25th of February; use the course portal page ("Elaborati" section) to upload it.

John has decided to throw a house party for his 30th birthday and he wants to go to the grocery store for shopping. He has a list of items and constraints he must respect during the shopping. This information is stored into a file where each line has the following format:

```
item weight value availability
```

where `item` is the name of the item, `weight` indicates the weight of the product, `value` is its price, and `availability` indicates the maximum availability of that product. The first line of the file indicates the total number of products (lines) within the file. Notice that `weight`, `value`, and `availability` are positive integers, and that `item` is a string of maximum 50 characters.

Write a program that helps John to fill his cart with products such that the **total weight** does not exceed `W`, each product count does not exceed its availability, and their **total price** (i.e., value) is maximized. The weight `W` and the name of the file are passed as parameters to the program. If a solution can be found, the program must print all items selected with the corresponding count, weight, and price, following the example reported below. If a solution cannot be found, the program should report an error message.

For example, given the input file

```
6
water      153   200   2
sandwich    50    60   2
banana      27    60   2
apple       39    40   3
cheese      23    30   1
beer        52    10   3
```

If the cart capacity is equal to $W = 400$, then the output would be:

```
          count weight price
water         2    306   400
banana        2     54   120
apple         1     39    40

solution      5    399   560
```