Reserved Cells

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

Algorithms and Programming

28 January 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 integers stored in an array:

   6 3 2 4 10 9 3 0 11 3 4 9 7 10 2

   sort it in ascending order using counting sort.

   Show the content of arrays A, B and C and all relevant intermediate steps on the array C.

   Show how counting sort can be implemented in C language, and specify its asymptotic complexity. Motivate the conclusions intuitively.

2. (2.0 **points**)
Given the following program:

```
1  #include <stdio.h>
2
3  void f (int l, int r) {
4    static int n_call = 0;
5
6    if (l>=r) {
7      return;
8    }
9
10   fprintf (stdout, "A: n_call=%d l=%d r=%d\n", n_call++, l+1, r);
11   f (l+1, r);
12   fprintf (stdout, "B: n_call=%d l=%d r=%d\n", n_call++, l+1, r-1);
13   f (l+1, r-1);
14   fprintf (stdout, "C: n_call=%d l=%d r=%d\n", n_call++, l, r-1);
15   f (l, r-1);
16
17   return;
18 }
19
20 int main () {
21   f (9, 11);
22 }
```

draw the recursive tree generated by function `f`. For each recursive call, report the values of all parameters and the exact output generated by the procedure. Express the asymptotic time complexity $T(n)$ of the procedure writing its **recurrence equation**.

3. (2.0 **points**)
10 **credit course** (01$OGDLP$)
Consider a binary tree with 13-nodes. Its visits return the following sequences:

pre-order:	12	9	13	11	2	1	6	7	3	5	15	4	10
in-order:	13	9	1	2	6	11	12	7	15	5	3	10	4
post-order:	13	1	6	2	11	9	15	5	10	4	3	7	12

Draw the original binary tree.

12 **credit course** (02$OGDLM$)
Given an initially empty BST, perform the following sequence of operations on the BST **root**:

$$+7 \quad +29 \quad +13 \quad +18 \quad +23 \quad +15 \quad +17 \quad -15 \quad -18$$

where each + indicates a root insertion and each − a node extraction. Report all relevant intermediate steps.

When this approach can be preferred to the one in which new nodes are inserted on the leaves?

4. **(2.0 points)**
   Given the sequence of keys $FARFROMHOME$, 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 23 where insertion of the previous sequence character by character occurs. Assume open addressing with double hashing ($h_1(k) = k\%23$, $h_2(k) = 1 + k\%97$). Show all relevant intermediate steps.

   Define what a **collision** is and how collisions can be managed.

   Define what the **load factor** is, and compute its value for the previous hash-table once the previous sequence of keys has been inserted.

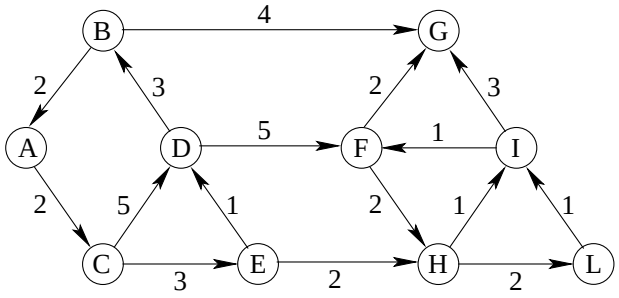
5. **(2.0 points)**
Suppose to have an initially empty priority queue implemented with a minimum heap. Given the following sequence of integers and $\star$ characters:

$$13 \quad 12 \quad 17 \quad 15 \quad 9 \quad 3 \quad 10 \quad 9 \quad 7 \quad * \quad * \quad 2 \quad *$$

where each integer corresponds to an insertion into the priority queue and each character $\star$ corresponds to an extraction, show the priority queue after each operation and return the sequence of values extracted.

Which library functions are usually used to manipulate a priority queue?

6. **(2.0 points)**
Given the following directed and weighted graph, find all shortest paths connecting node A with all the other nodes resorting to Dijkstra's algorithm.



If necessary, consider nodes and edges in alphabetical order.

Would Bellman-Ford's algorithm find the same result? Justify your answer.

Algorithms and Programming
28 January 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 Friday the 31th of January; use the course portal page ("Elaborati" section), and follow the appropriate procedure, to upload it.

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

```
void invert_sequence (int *v1, int n, int *v2);
```

which receives two arrays of integers `v1` and `v2` and their size `n`, and it stores in `v2` the same values stored in `v1` but such that each ascending sub-sequence is reversed (i.e., each ascending sequence becomes a descending one).

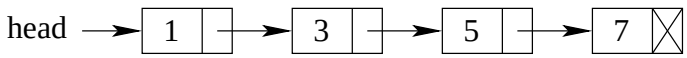
For example, if
`v1 = { 1, 2, 3, 4, 5, 0, 12, 13, 14, 2 }`,
then the function must generate
`v2 = { 5, 4, 3, 2, 1, 14, 13, 12, 0, 2 }`,

2 (4.0 **points**)
Write the function:

```
void int2list (int n, struct node **head);
```

which receives a positive integer `n` in input and it generates a list, referenced by `head`, including one element for each digit of the number `n`. When the function is called the pointer `head` is equal to `NULL`. Elements in the list must have the same order of the digits in the number `n`.

For example, if `n = 1357`, the function must create the following list:



3 (6.0 **points**)
Write the function:

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

which receives a string `str`, and it prints all possible palindromic partitions of that string, i.e., all partitions of `str` into a set of palindromic sub-strings.

For example if
`str = "annabella"`
the function must print the following 6 partitions:

```
a-n-n-a-b-e-l-l-a
a-nn-a-b-e-l-l-a
a-n-n-a-b-e-ll-a
a-nn-a-b-e-ll-a
anna-b-e-l-l-a
anna-b-e-ll-a
```

Algorithms and Programming
28 January 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 Friday the 31th of January; use the course portal page ("Elaborati" section), and follow the appropriate procedure, to upload it.

Given a Boolean formula, the Boolean Satisfiability problem (often abbreviated with SAT) is the problem of determining if there is an assignment to the variables that satisfies the formula.
Write a program able to solve this problem, with the following specifications.

A formula is expressed as a conjunction (AND) of one or more clauses, and each clause is a disjunction (OR) of literals (variables). In other words, a formula is expressed as a Boolean AND of Boolean ORs.
An example of such a formula, is the following one:

$$f \quad = \quad (x_1 + x_2 + x_3) \cdot (x_2 + \overline{x_3}) \cdot (x_3 + \overline{x_1}) \cdot (x_1 + \overline{x_3})$$

In the formula, symbols + indicates Boolean OR, symbols $\cdot$ indicates Boolean AND, and overlined symbols represent the negation NOT of the corresponding literal. If, for example, we make the following assignments $\{x_1 = 0, x_2 = 1, x_3 = 0\}$ the formula evaluates to true (i.e., it is satisfied or SAT), as

$$f \quad = \quad (0 + 1 + 0) \cdot (1 + \overline{0}) \cdot (0 + \overline{0}) \cdot (0 + \overline{0}) \quad = \quad 1 \cdot (1 + 1) \cdot (0 + 1) \cdot (0 + 1) \quad = \quad 1 \cdot 1 \cdot 1 \cdot 1 \quad = \quad 1$$

Obviously any formula can be satisfied by more than one assignment to its variables. For example, the previous formula is also satisfied when $\{x_1 = 0, x_2 = 1, x_3 = 0\}$ and when $\{x_1 = 1, x_2 = 1, x_3 = 1\}$.

Given a formula with n clauses and m variables, it is possible to store it in a file as a matrix including n rows of m columns of integer values. For example, the previous formula can be stored as:

```
4 3
3 1 2 3
2 2 -3
2 3 -1
2 1 -3
```

where:

- The first line indicates the number of clauses $n = 4$ and the total number of variables $m = 3$.
- Each clause is stored into a single line of the file. The first number on each line indicates the number of variables in the clause. Each positive number represents a variable. Each negative one the negation (NOT) of that variable.

Thus, the second row "`3 1 2 3`" indicates the clause $(x_1 + x_2 + x_3)$, and the last one "`2 1 -3`" the clause $(x_1 + \overline{x_3})$.

Write a program able to:

- Receive the name of a file storing the formula on the command line.
- Store the formula in a proper data structure.
- Search for a possible mapping between the formula variables and the Boolean values 0 and 1 which satisfies the formula. Print this mapping if it exists. State that the formula is UNSAT (i.e., not satisfied) if it does not exist.