Reserved Cells

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

Algorithms and Programming

22 February 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**)
   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-union, 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.

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

2. (2.0 **points**)

10 **credit course** ($01OGDLP$)

Convert the following expression from in-fix to pre-fix notation (Polish Notation) and from in-fix to post-fix notation (Reverse Polish Notation).

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

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

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

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

$$[1,7] \quad [14,16] \quad [4,5] \quad [10,12] \quad [8,13] \quad [2,8] \quad [11,13] \quad [12,20] \quad [6,9] \quad [0,3] \quad [15,23]$$

After that, search in the resulting Interval BST an intersection with intervals $[12,13]$, and $[19,26]$. Describe all recursion steps performed to search these intervals, and specify when it is necessary to recur on the right child and when to recur on the left one.

3. **(2.0 point)**
Given the following sequence of integers stored into an array:

12 4 7 0 8 15 6 14 3 19 6 11 10 13 9

perform the first 2 steps of quick sort to sort the array in ascending order. At each step indicate the pivot element you have selected.

Steps must be improperly considered in breadth on the recursion tree, rather than in depth. Return as a result the two partitions of the original array and the two partitions of those found at the previous step.

4. **(2.0 points)**
Given the following sequence of integers stored into an array:

1 19 7 11 21 3 9 17 5 4 11 10 5 12 8

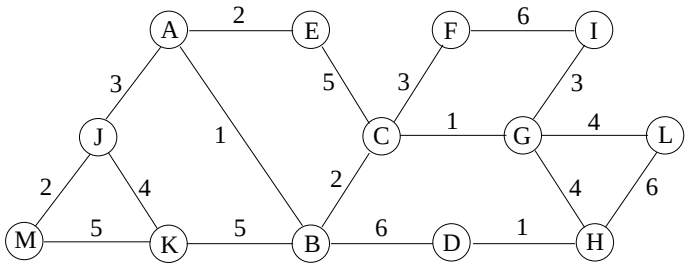
turn it into a heap, assuming to use an array as underlying data structure. Draw each step of the heap-building process, as well as the final result. Assume that, at the end, the largest value is stored at the heap's root. Execute the first three steps of the heap sort algorithm on the heap built at the previous step.

Show all relevant intermediate steps.

5. (2.0 **points**)

10 **credit course** ($01OGDLP$)

Given the following undirected and weighted graph find a minimum spanning tree using Kruskal's algorithm.



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", a "crossing edge", and a "set of edges respecting a cut" are. Report an example to illustrate these three concepts.

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

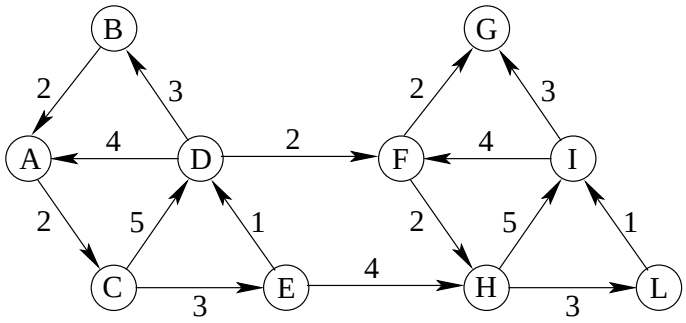
Given the following activity set, where the i-th activity is identified by the pair $[s_i, f_i)$ where s_i is the starting time and f_i is the finishing time:

Activity	s_i	f_i
P_1	0	3
P_2	4	8
P_3	11	17
P_4	14	19
P_5	8	11
P_6	3	9
P_7	19	21
P_8	23	27
P_9	7	12
P_{10}	2	10

Using a greedy algorithm, find the largest subset of mutually compatible activities.

Explain what "greedy algorithms" are, and provide a few examples of greedy algorithms achieving an optimal solution.

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
22 February 2019
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 Wednesday the 27th of February; use the course portal page ("Elaborati" section) to upload it.

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

```
void matrix_check (int **mat, int r, int c, int **M);
```

which receives an integer matrix `mat` of `r` rows and `c` columns, and a matrix `M` of size (3×3), and it finds all possible overlapping positions of `M` with respect to `mat` such that no elements of `mat` different from zero overlap with an element of `M` equal to zero. For each possible overlapping position, the function must print the coordinates of the top-left overlapping element of `mat`. The candidate must consider only overlapping positions where `M` is completely included within `mat` boundaries. Suppose that both `r` and `c` are larger than or equal to 3.
For example, if `mat` and `M` are the following ones:

$$\text{mat} = \begin{array}{|ccccc|}\hline 8 & 6 & 8 & 0 & 0 \\ \mathbf{0} & 5 & 0 & 2 & 3 \\ 1 & 4 & \mathbf{0} & 1 & 0 \\ 0 & 3 & 0 & 5 & 8 \\ 7 & 1 & 0 & 0 & 0 \\ \hline \end{array} \qquad \text{M} = \begin{array}{|ccc|}\hline 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \\ \hline \end{array}$$

the function must print coordinate pairs $(1, 0)$ and $(2, 2)$, representing the bold elements.

2 (4.0 **points**)
A list, with elements of type `list1_t`, stores pairs of integers {`value, frequency`}. Write a function able to "expand" such a list, i.e., to create a completely new list, with elements of type `list2_t`, in which for each element {`value, frequency`} in the first list there are `frequency` elements of value `value` in the second list. The second list must be sorted. The function prototype must be

```
list2_t *list_expand (list1_t *p1);
```

where `p1` is a pointer to the head of the first list, and the function returns the pointer to the second list. The candidate must report the C data structures used to represent elements `list1_t` and `list2_t`.
For example, let us suppose that `p1` points to the list $\{(3, 5), (4, 3), (2, 1), (6, 2)\}$. The function has to create the following list $\{2, 3, 3, 3, 3, 3, 4, 4, 4, 6, 6\}$ and to return its head pointer.

3 (6.0 **points**)
A vending machine can distribute change using only coins of a given set of denominations. For each denomination the vending machine has a limited supply of coins.
Write the function:

```
void change_making(float *bold, int *coins, int n, float change);
```

¡ that receives an array *vals* of n coin denominations (floating point values), an array *coins* of n integers (each representing the number of available coins for a given denomination) and a floating point value *change*, representing the amount of change due. The function has to print which coins the vending machine should return to the user in order to make up the given change using the minimum number of coins. If the change cannot be returned using the current supply of coins the function has to print an error message.
For instance, suppose that the vending machine may distribute change using only the following coin denominations: $vals = \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1\}$, for which it holds the following supply $coins = \{20, 12, 1, 0, 9, 4, 7\}$. Given an amount of change equal to 1.15, the function must produce an output similar to the following: $\{0.5, 0.2, 0.2, 0.2, 0.05\}$.

Algorithms and Programming
22 February 2019
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 Wednesday the 27th of February; use the course portal page ("Elaborati" section) to upload it.

The Smithsonian Institution in Washington, is the world's largest museum and research complex, with 19 museums and 9 research centers. Each museum can be visited for free, following specific time slots pre-defined during the day, each of which with a maximum number of visitors. The candidate must write an application to manage museum reservations.

A file specifies the characteristics of museums, such as:

```
National_Museum_of_Natural_History 5
10.00-11.30 1780
11.30-13.00 1780
13.00-14.30 1830
14.30-16.00 1600
16.00-17.30 1400
Smithsonian_National_Air_and_Space_Museum 7
10.00-11.00 650
11.00-12.00 680
...
```

where each museum is identified by its name (a string of at most 100 characters), followed by the number of visiting time slots. Each time slot is then specified by a distinct line following the museum specification. Each of these lines contains the starting and finishing time of the slot, followed by the maximum number of people allowed in the museum during that slot.

The application has to read the file to store the information into a proper data structure, and then it has to implement a suitable set of functions to perform the following operations. The data structure used should be designed in order to enable an efficient execution of these operations optimizing memory usage at the same time.

- Make a new reservation under a given visitor's name for a specific museum, day, time slot and a certain number of people. Reservations can be made for any given day in the current year, numbering days from 0 (January 1st) to 364 (December 31st). The application must accept a new reservation only if the number of visitors does not exceed the maximum number of people allowed in the museum in that time slot for that day. A reservation must be refused (printing an error message) if the number of visitors exceeds such a threshold or if the reservation contains any malformed data. For instance, given the following input:

  ```
  National_Museum_of_Natural_History, 23, 13.00-14.30, 3, Smithson_James
  ```

  the application should reserve (if possible) a visit to the `National_Museum_of_Natural_History`, for the day number `23`, in the time slot `13.00-14.30`, for `3` people, under the name of `Smithson_James`.
- Print all reservations done by a visitor given the visitors' name. For each reservation, the application must print the name of the museum, the day, time slot, and number of people in the group. The application must print an error message in case no reservation is found under the given visitor's name. The average time complexity to perform such an operation must be at worst logarithmic in the number of visitors. For instance, given the visitor's name `Smithson_James`, the application should print all reservations done by `Smithson_James` if any reservation has been done, such as:

  ```
  National_Museum_of_Natural_History, 23, 13.00-14.30, 3
  Smithsonian_National_Air_and_Space_Museum, 24, 10.00-11.00, 2
  ...
  ```