Algorithms and Data Structures

27.01.2022

Exercise 1 (1.0 points)
Given the following array of integer values, perform the first step of quicksort to sort the array in ascending order, thus from the initial array generate the right and the left partitions.

```
 1 10  6  4  8  5  2  7 11  9
```

Report 3 integer values: The pivot selected on the original array, the pivot you would select on the left partition generated from the original array, and the pivot you would select on the right partition generated (again) from the original array. No other symbols must be included in the response. This is an example of response format: 13 1 10

Exercise 2 (4.0 points)
Suppose an array v stores N integer values (v[N]) and we look for a value key in the array, using the following four functions:

- A sequential linear search with the array being unordered.
- A sequential linear search with the array being sorted.
- A binary sequential search with the array being sorted.
- A binary recursive search with the array being sorted.

Report the code for the recursive search and illustrate how to compute the asymptotic costs of the fopur functions. Which is their final asymptotic complexity? Motivate your answers and give the intuition behind the asymptoticcosts.

Exercise 3 (1.0 points)
Consider a binary tree whose visits return the following sequences.

```
Pre-order:      A    B    C    F    D    G    L    E    H    I
In-order:       B    F    C    A    G    L    D    H    I    E
Post-order:     F    C    B    L    G    I    H    E    D    A
```

Report the sequence of keys stored on the leaves of the tree, moving on the tree from left to right. Please, report the list of keys on the same line, separated by a single space. No other symbols must be included in the response. This is an example of the response format: A X C Y etc.

Exercise 4 (1.0 points)
Given an initially empty BST, perform the following sequence of operations on the BST leaves. Each positive value indicates a leaf insertion and each negative value a node extraction.

```
   4    6   10   13    7   18   20    3    9   16
```

Report the key of all leaves of the final BST considering them from left to right. Please, report all nodes' key as a sequence of integer values separated by a single space. No other symbols must be included in the response. This is an example of the response format: 23 4 5 3 etc.

Exercise 5 (1.0 points)
Insert the following sequence of keys into an initially empty hash table. The hash table has a size equal to `M=23`. Insertions occur character by character using open addressing with double hashing. Use functions $h_1(k) = k\%M$ and $h_2(k) = 1 + (k\%97)$. Each character is identified by its index in the English alphabet (i.e., `A=1`, ..., `Z=26`). Equal letters are identified by a different subscript (i.e., `A` and `A` become `A1` and `A2`).

```
A L L A
```

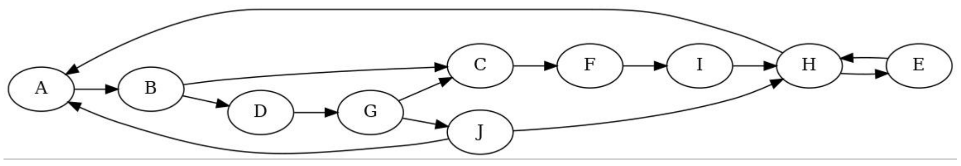
Indicate in which elements are placed the last two letters of the sequence, i.e., `L` and `A`, in this order. Please, report your response as a sequence of integer values separated by one single space. No other symbols must be included in the response. This is an example of the response format: 3 4

Exercise 6 (1.0 points)
Given the following sequence of integers stored into an array, turn it into a heap, assuming to use an array as an underlying data structure. Assume that, in the end, the largest value is stored at the heap's root. Then, execute the first two steps of heapsort on the heap built at the previous step.

```
 7  8 10  3  5  6  2 13
```

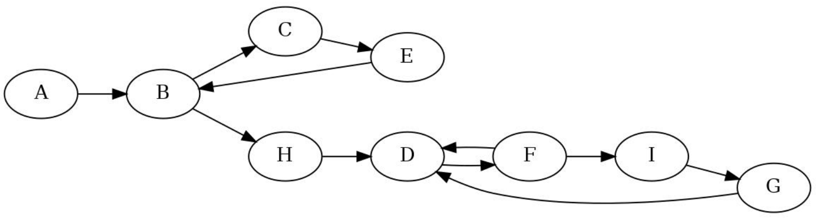
Report the final content of the entire array at the end of the above process. Please, show the entire content of the array as a sequence of integer values separated by a single space. No other symbols must be included in the response. This is an example of the response: 0 3 2 6 8 etc.

Exercise 7 (1.0 points)
Visit the following graph in breadth-first, starting at node A. When necessary, consider nodes and edges in alphabetic order.


$$\begin{tikzcd}
A \arrow[r] & B \arrow[rrr] \arrow[r] & D \arrow[r] & G \arrow[r] \arrow[r] & C \arrow[r] & F \arrow[r] & I \arrow[r] & H \arrow[r, shift right] \arrow[lllllll, bend right] & E \arrow[l, shift right] \\
& & & & J \arrow[rrru] \arrow[llllu, bend left] & & & &
\end{tikzcd}$$

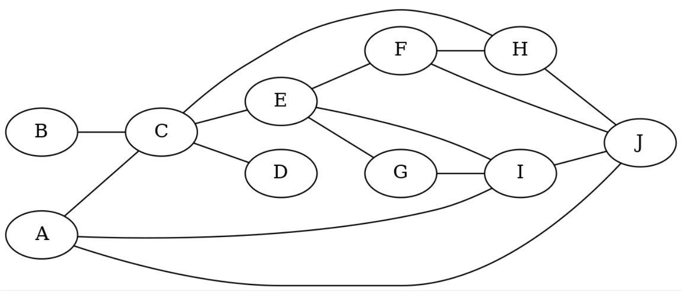
Display the minimum distance of all vertices from the starting one. Please, indicate the distance of all vertices sorted in alphabetic order (i.e, display the distance for A B C D, etc.). Report a sequence of integer values separated by one single space. No other symbols must be included in the response. This is an example of the response: 0 3 2 6 8 etc.

Exercise 8 (1.0 points)
Visit the following graph depth-first, starting at node A. Label nodes with discovery and end-processing times. Start with the discovery time set to 1 on A. When necessary, consider nodes and edges in alphabetic order.



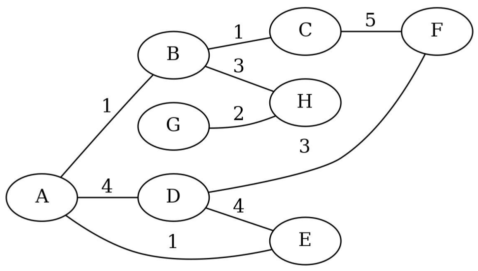
Display the end-processing time of all vertices. Please, indicate the end-processing time of all vertices sorted in alphabetic order (i.e, display the end-processing time for A B C D etc.). Report a sequence of integer values separated by one single space. No other symbols must be included in the response. This is an example of the response: 15 13 2 16 8 etc.

Exercise 9 (1.0 points)
Given the following graph find all articulation points. If necessary, consider nodes and edges in alphabetical order.



Display all articulation points alphabetically. Please, report only the list of vertices separated by a single space. No other symbols must be included in the response. This is an example of the response format: A B C etc.

Exercise 10 (1.0 points)
Given the following undirected and weighted graph find a minimum spanning tree using Kruskal algorithm.



Indicate the total weight of the final minimum spanning tree. Report one single integer value. No other symbols must be included in the response. This is an example of the response format: 13

Exercise 11 (2.0 points)

Analyze the following program and indicate the exact output generated.

```
void f (char *);
int main(void) {
  char *s = "This 12345 Is A String 678";
  char *d;
  d = strdup (s);
  f (d);
  fprintf (stdout, "%s", d);
  return (1);
}
void f (char *s) {
  int i, j;
  i = 0;
  while (i < strlen(s)) {
    if (s[i]==' ' || (s[i]>='0' && s[i]<='9') || (s[i]>='A' && s[i]<='Z')) {
      for (j=i+1; j<strlen(s)+1; j++) s[j-1] = s[j];
    } else {
      i = i + 1;
    }
  }
  return;
}
```

Report the exact program output with no extra symbols.

Exercise 12 (2.0 points)

The following data structure specifies a list of element struct m in which each element individuate a list of element struct p. The function visits the list of lists, displaying all data fields.

```
typedef struct m m_t;
typedef struct p p_t;
struct m {
  char *name;
  char *id;
  p_t *p;
  m_t *next;
};
struct p {
  char *name;
  int price;
  p_t *next;
};
void printAll (
  m_t *head
  )
{
  m_t *tmp1;
  p_t *tmp2;
  tmp1 = head;
  while (tmp1 != NULL) {
    fprintf (stdout, "%s %s\n", tmp1->name, tmp1->id);
    tmp2 = tmp1->p;
    while (tmp2 != NULL) {
      fprintf (stdout, "  - %s %d\n", tmp2->name, tmp2->price);
      tmp2 = tmp2 ->next;
    }
    tmp1 = tmp1->next;
  }
  return;
}
```

Indicates which ones of the following statements are correct. Note that incorrect answers imply a penalty in the final score.

1. The line `tmp2 = tmp1− >next;` must to be `tmp2 = tmp1− >p;`.
2. The variable head must be passed by reference to function `printAll`, i.e., as $m_t **head$.
3. The function has a cost that is linear in the number of elements in the main list.
4. A function looking for a specific element in a secondary list has the same asymptotic complexity of function printAll.
5. The function has a cost that is linear in the total number of elements (main plus secondary lists).
6. The line `tmp2 = tmp2− >next;` must to be `tmp2 = tmp2− >p;`.

Exercise 13 (2.0 points)
Analyze the following recursive program.

```
void f1 (int n);
void f2 (int n);
void f3 (int n);
void f1 (int n) {
  if (n<=0) {
    return;
  }
  printf ("1");
  f2 (n-1);
  return;
}
void f2 (int n) {
  if (n<=0) {
    return;
  }
  printf ("2");
  f3 (n+1);
  return;
}
void f3 (int n) {
  if (n<=0) {
    return;
  }
  printf ("3");
  f1 (n-1);
  return;
}
int main () {
  fprintf (stdout, "f1(5): ");
  f1(5);
  return 1;
}
```

Report the exact program output with no other symbols.

Exercise 14 (3.0 points)
Write the function

```
void invert_string (char *s1, char **s2);
```

which receives an input string (of unknown length) s1 and generates an output string (initially, a NULL pointer) s2. The string s1 includes only uppercase letters. The function allocates the string s2, and then stores into it the same letters stored in s1 but such that all strictly ascending sub-sequences of single characters appearing in s1 are transformed into strictly descending sub-sequences of single characters into s2. The string s2 must be dynamically allocated by the function. For example, if s1 is the following:

```
A   B   C   D   D   D   D   Z   Y   X   W   E   F   G
```

the function has to store into s2 the following string:

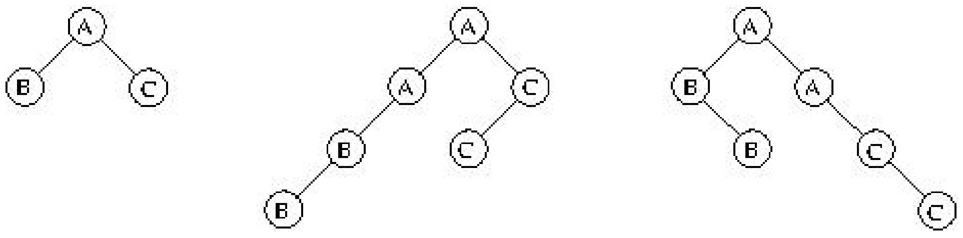
```
D   C   B   A   D   D   Z   D   Y   X   W    G   F   E
```

as the sequence `''A B C D''` is strictly ascending and must be transformed into `''D C B A''` and the same consideration holds for `''D Z''` transformed into `''Z D''`, etc.
Write the entire program using standard C libraries but implement all required personal libraries. Modularize the program adequately, and report a brief description of the data structure and the logic adopted in plain English. Unclear or awkward programs, complex or impossible to understand, will be penalized in terms of the final evaluation.

Exercise 15 (5.0 points)
Write the function

```
void double_tree (struct node *root, char flag);
```

that for each node in a binary tree creates a new duplicate node, and inserts the duplicate as the left child of the original node if flag='L', and as the right child of the original node if flag='R'. Report the C structure (struct node) storing string keys ('A', 'B', 'C', etc. must be considered as dynamic strings, as they are characters just for the sake of simplicity).
For example, the tree on the left-hand side of the following picture must be transformed into the one in the middle if flag='L', and into the one on the right in flag='R'. Write the entire program using standard C libraries but implement

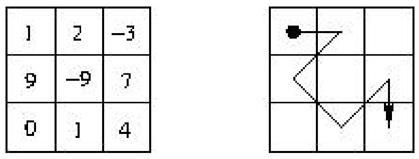


all required personal libraries. Modularize the program adequately, and report a brief description of the data structure and the logic adopted in plain English. Unclear or awkward programs, complex or impossible to understand, will be penalized in terms of the final evaluation.

Exercise 16 (9.0 points)
A map is represented as a matrix of integer values. The starting cell of the map is always at the top-left corner (element `[0,0]`), while the ending cell is always at the bottom-right corner (element `[r-1, c-1]`, where `r` and `c` are the numbers of rows and columns of the matrix, respectively). It is possible to walk (i.e., visit) the map by moving from one cell to any adjacent element (8 at most), but each cell has to be visited only once. Write the recursive the C function matVisit that is able to find a path from the starting point to the ending point whose sum of the content of the visited cell is maximum. The function receives four parameters: the matrix `mat`, its number of rows `r`, its number of rows `c`, and a flag `f`:

```
void mat_visit (int **mat, int r, int c, int f);
```

If there are more paths with the same (maximum) sum, the function has to display the shortest path if `f=0` and the longest one if `f=1`. The desired path has to be displayed with a format similar to the one reported in the example. The following figure illustrates a matrix mat, with r=2 and c=2 (left-hand side), and the corresponding solution (right-hand side).



The indicated path should be reported with a format similar to the following one: `[0,0]1 - [0,1]2 - [1,0]9 - [2,1]1 - [1,2]7 - [2,2]4 - sum = 24`
Write the entire program using standard C libraries but implement all required personal libraries. Modularize the program adequately, and report a brief description of the data structure and the logic adopted in plain English. Unclear or awkward programs, complex or impossible to understand, will be penalized in terms of the final evaluation.