Algorithms and Data Structures

07.02.2023

Exercise 1 (1.5 points)
Given the following array of integer values, perform the first step of quicksort to sort the array in ascending order; then, from the initial array, generate the right and the left partitions.

```
10  9 11  4  1  4  2  3  7  5
```

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 the response format: 13 1 10

Exercise 2 (1.5 points)
Consider a binary tree whose visits return the following sequences.

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

Report the sequence of keys stored on the tree leaves, 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 3 (1.5 points)
Insert the following sequence of keys into an initially empty hash table. The hash table has a size equal to `M=19`. Insertions occur character by character using open addressing with quadratic probing (with $c_1 = 1$ and $c_2 = 1$). 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`).

```
R I C C I A
```

Indicate in which elements are placed the last three letters of the sequence, i.e., C, I, 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 11

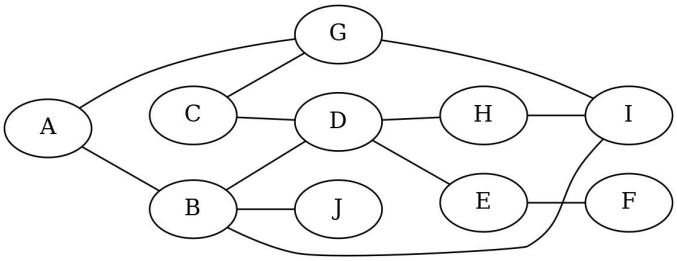
Exercise 4 (1.5 points)
The following capital letters are given with their absolute frequency.

```
A:4 B:7 C:8 D:10 E:17 F:5
```

Find an optimal Huffman code for all symbols in the set using a greedy algorithm. Indicate which one of the following encoding is correct.

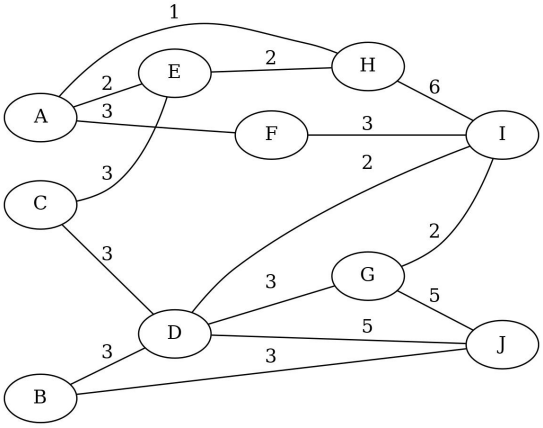
1. A(01) B(10) C(110) D(1111) E(1110) F(00001)
2. A(010) B(10) C(110) D(1111) E(1110) F(00)
3. A(01) B(100) C(110) D(1111) E(1110) F(00)
4. A(000) B(100) C(101) D(01) E(11) F(001)
5. A(11) B(10) C(110) D(111) E(1110) F(00)
6. A(101) B(110) C(11) D(1111) E(1110) F(00)

Exercise 5 (1.5 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 6 (1.5 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 7 (4.0 points)
Define what a Minimum Spanning Tree is and specify the differences from a Single Source Shortest Path problem. Define what a cut, a light edge, and a safe edge are. Clarify why we define safe edges and how the algorithms of Prim and Kruskal use this concept. Are they greedy algorithms? Do they find an optimal solution? There is only one or more than one solutions to the MST problem?

Exercise 8 (2.0 points)
Analyze the following program and indicate the exact output it generates. Please, report the exact program output with no extra symbols.

```
#define R 3
#define C 5
#define D 3
void f (int [][C]);
int main(void) {
  int mat[R][C] = {
    {1,2,3,4,5},
    {1,2,3,4,5},
```

```
    {1,2,3,4,5},
  };
  f (mat);
  return (1);
}
void f (int mat[][C]) {
  int i1, i2, j1, j2, s;
  for (i1=0; i1<=R-D; i1++) {
    for (j1=0; j1<=C-D; j1++) {
      s = 0;
      for (i2=0; i2<D; i2++) {
        for (j2=0; j2<D; j2++) {
          s = s + mat[i1+i2][j1+j2];
        }
      }
      printf ("%d ", s);
    }
  }
 return;
}
```

Exercise 9 (2.0 points)

The following function allocates a three-dimensional matrix (a two-dimensional matrix of strings) and reads the strings from standard input. The parameters `r` and `c` are the numbers of rows and columns of the matrix.

```
char ***read (int *r, int *r) {
  char word[MAX_LEN], ***mx;   // LINE 1
  int i, j;
  LINE 2
  if (mx == NULL) {
    printf("Memory allocation error.\n");
    exit(EXIT_FAILURE);
  }
  for (i=0; i<r; i++) {
    mx[i] = (char **)malloc(c * sizeof(char *));
    if (mx[i] == NULL) {
      printf("Memory allocation error.\n");
      exit(EXIT_FAILURE);
    }
  }
  for (i=0; i<r; i++) {
    for (j=0; j<c; j++) {
      printf("String: ");
      fscanf("%s", word);
      LINE 3
      if (mx[i][j] == NULL) {
        printf("Memory allocation error.\n");
        exit(EXIT_FAILURE);
      }
    }
  }
  return mx;
}
```

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

1. In LINE 1 the array word can be defined as: `char word[r].`
2. In LINE 1 the array word must be either defined statically or allocated dynamically.
3. LINE 2 can be: `mx = malloc(r * sizeof(char **)).`
4. LINE 3 can be: `mx[i][j] = strdup (word).`
5. LINE 2 can be: `mx = (char ***)malloc(r * sizeof(char **)).`
6. LINE 2 can be: `mx = (char **)malloc(r * sizeof(char *)).`
7. LINE 3 can be: `strcpy (mx[i][j], word).`
8. LINE 2 can be: `mx = malloc(r * sizeof(char *)).`

Exercise 10 (2.0 points)

Analyze the following program and indicate the exact output it generates. Please, report the exact program output with no extra symbols.

```
void f (int);
void fp (int);
void fm (int);
void f (int n) {
  if (n%2==0) {
    fp (n);
  } else {
    fm (n);
  }
  return;
}
void fp (int n) {
  if (n<=0) {
    return;
  }
  printf ("+");
  f (n-1);
  return;
}
void fm (int n) {
  if (n<=0) {
    return;
  }
  printf ("-");
  f (n-1);
  return;
}
int main () {
  f(10);
  return 1;
}
```

Exercise 11 (3.0 points)
A square matrix m of size n stores only capital alphabetic characters. Write the function

```
void check (char **m, int n);
```

which receives the matrix `m`, its size `n`, and displays the row including the same alphabetic characters more times. The function displays nothing if no row contain the same character more than once. The function displays a row at choice if more rows contain a character the same number of times.
For example, if the matrix is the following one (with size `n=6`):

```
X A E I O U
U X A E I O
O U X A E I
I O U X A E
E I O U X A
X Y Z X Y X
```

the last row includes three letters `X`, thus the function must display that row, i.e., `5`.

```
X Y Z X Y X
```

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 12 (5.0 points)
A file stores an undefined number of rows. Each row includes two fields: An integer value, and a string of at most 100 characters. Those fields are separated by one or more spaces.

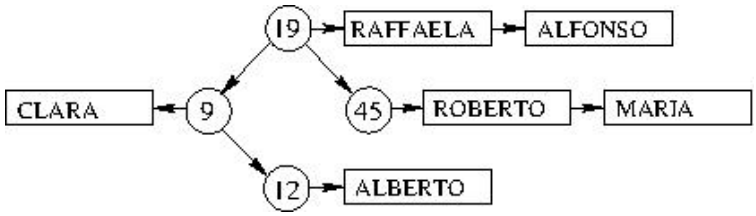
Write a C function able to organize the file content into a BST in which each node points to a list. The keys of the BST are integer values and the lists store strings. More.texifically, all record of the file with the same integer value must be stored into the list of the same BST element.
Write the function:

```
bst_t *insert (char *name);
```

which receives the file name and returns the pointer to such a BST. The lists must not be sorted and may contain repeated elements.
Define the type bst_t for the BST nodes, and the type list_t for the list nodes. The string within the type list_t must be dynamically allocated.
For example, if we suppose the file includes the following rows

```
19 ALFONSO
45 MARIA
9 CLARA
19 RAFFAELA
45 ROBERTO
12 ALBERTO
```

the data structure created by the function must be the following one.



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 13 (9.0 points)
Write a recursive function

```
void generate (char *name, int n);
```

which stores in the file name all decimal integers of n digits for which:

- the digits of weights 10^0, 10^2, 10^4, 10^6, etc., have an even value (e.g., 0, 2, 4, 6, 8)
- the digits of weights 10^1, 10^3, 10^5, 10^7, etc., have an odd value (e.g., 1, 3, 5, 7, 9)
- the sum of all even digits equals the sum of all odd digits.

For example, with `n=4` digits, the following numbers satisfy specified properties: 1430 (with sum $0+4=3+1=4$), 3652 (with sum $2+6=5+3=8$), 5676 (with sum $6+6=7+5=12$).
The solution must prune the recursion tree as soon as possible, i.e., solutions that generate all integer numbers of `n` digits and display only the numbers satisfying the requested properties will be penalized in terms of the final mark.
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.