Algorithms and Data Structures

17.05.2023

Exercise 1 (1.5 points)
Given the following array of integer values, sort it in descending order using the merge sort procedure.

```
 8  5  6  7 13  3 12  2 17 11
```

Display the content of the array just before the last (and conclusive) merge step (the one delivering the final and sorted array). 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.

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

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

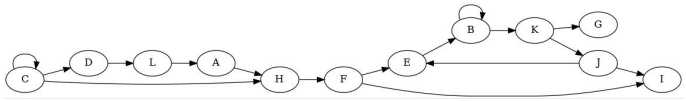
Report the sequence of keys stored on the tree leaves, moving on the tree from left to right. Please, report the list of keys on the same line, separated by a single space. No other symbols must be included in the response. This is an example of the response format: A X C Y etc.

Exercise 3 (4.0 points)
Suppose a BST stores strings of unknown size. Each BST node has a standard structure but it also includes an integer value storing the size of the BST rooted at that node. Write the function to insert a new key into the leaves of the BST. The node type and the function prototypes are the following.

```
typedef struct node_s {
  char *key;
  int size;
  struct node_s *left;
  struct node_s *right;
} note_t;
node_t *insert (node_t *root, char *key);
```

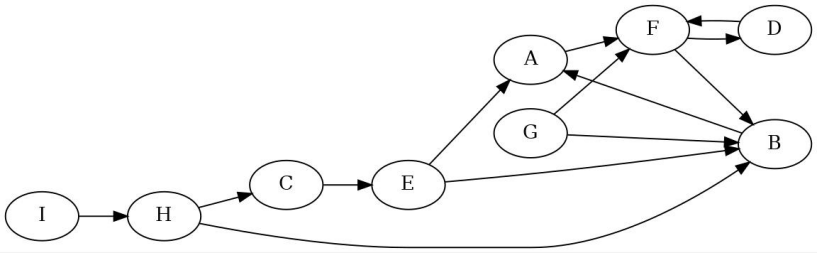
Pay particular attention to how to update the size of all visited nodes.

Exercise 4 (1.5 points)
Visit the following graph in breadth-first, starting at node C. When necessary, consider nodes and edges in alphabetic order.



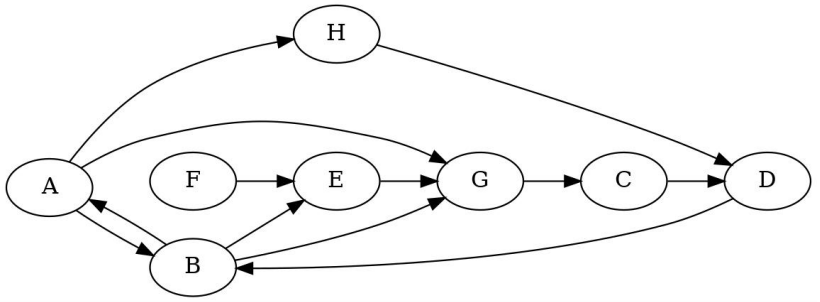
Display the minimum distance of all vertices from the starting one. Please, indicate the distance of all vertices sorted in alphabetic order (i.e, display the distance for A B C D, etc.). Report a sequence of integer values separated by one single space. No other symbols must be included in the response.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

Exercise 5 (1.5 points)
Visit the following graph in depth-first, starting at node I. Label each edge as tree (T), back (B), forward (F), and cross (C). When necessary, consider nodes and edges in alphabetic order.



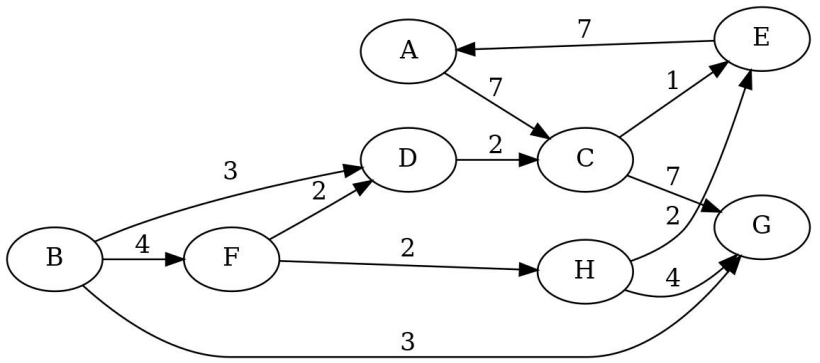
Report the label of the edges CE, GB, and GF in this order. Please, indicate the edge type of these 3 arcs with a single letter, i.e., T, F, B, C, separated by single spaces. No other symbols must be included in the response. This is an example of the response: F T B

Exercise 6 (1.5 points)
Given the following directed graph, represent the reverse graph and find all strongly connected components.



Compute all strongly connected components in the graph. Report them in alphabetic order and within each component indicates vertices in alphabetic order (i.e., ACZ and not CAZ or ZAC). Separate components with a single space. No other symbols must be included in the response. This is an example of the response format: ABC EF XY etc.

Exercise 7 (1.5 points)
Given the following directed and weighted graph, apply Dijkstra's algorithm to find all shortest paths connecting node B with all the other nodes. When necessary, consider nodes and edges in alphabetical order.



Report the shortest paths to all vertices. Please, indicate the shortest path to all vertices sorted in alphabetic order (i.e, display the shortest path 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 (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 MAX 32
void f (char s1[], char s2[], char s3[]);
int main(void) {
  char s1[] = "010101";
  char s2[] = "011101";
  char s3[] = "xxxxxx";
  f (s1, s2, s3);
  printf ("%s", s3);
  return (1);
}
void f (char s1[], char s2[], char s3[]) {
  int i, a, b, c, s, l;
  l = strlen(s1) > strlen(s2) ? strlen(s1) : strlen(s2);
  s3[l] = '\0';
  c = 0;
  for (i=l-1; i>=0; i--) {
    a = s1[i] - '0';
    b = s2[i] - '0';
    s = a + b + c;
    s3[i] = s%2 + '0';
    c = s/2;
  }
  return;
}
```

Exercise 9 (2.0 points)

The following function stores two files into a list of lists.

```
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;
};
m_t *load (char *filename1, char *filename2) {
  m_t *head, *new1;
  p_t *new2;
  FILE *fp;
  int price, found;
  char str1[MAX], str2[MAX];
  head = NULL;
  fp = fopen (filename1, "r");
  while (fscanf (fp, "%s%s", str1, str2) != EOF) {
    new1 = (m_t *) malloc (sizeof (m_t));
    if (new1==NULL) {
      fprintf (stdout, "Allocation error.\n");
      exit (1);
    }
    new1->name = strdup (str1);
    new1->id = strdup (str2);
    new1->p = NULL;
    new1->next = head;
    head = new1;
  }
  fclose(fp);
  fp = fopen (filename2, "r");
  while (fscanf (fp, "%s%s%d", str1, str2, &price) != EOF) {
    new2 = (p_t *) malloc (sizeof (p_t));
    if (new2==NULL) {
```

```
      fprintf (stdout, "Allocation error.\n");
      exit (1);
    }
    new2->name = strdup (str2);
    new2->price = price;
    found = 0;
    new1 = head;
    while (new1!=NULL && found==0) {
      if (strcmp (new1->id, str1)==0) {
        found = 1;
      } else {
        new1 = new1->next;
      }
    }
    if (found==1) {
      new2->next = new1->p;
      new1->p = new2;
    } else {
      fprintf (stderr, "There is no such a brand ...  Rule-out model\n");
    }
  }
  fclose(fp);
  return head;
}
```

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

1.
2.
3.
4.
5.
6.

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 i, int j, int k) {
  if (i<0 && j<0 && k<0) {
    return;
  }
  if (i>=0) {
    printf ("I");
    f (i-1, j, k);
  } else {
    if (j>=0) {
      printf ("J");
      f (i, j-1, k);
    } else {
      if (k>=0) {
        printf ("K");
        f (i, j, k-1);
      }
    }
  }
  return;
}
int main () {
  f(5,4,3);
 return 1;
}
```

Exercise 11 (3.0 points)

A square matrix m of size n stores integer values.
Write the function

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

which receives the matrix `m`, its size `n`, and a value `k` (smaller than `n`) and it displays the submatrix of `m` of size `k` whose sum of the element is the largest, and the submatrix of `m` of size `k` whose sum of the element is the smallest.
For example, if $n = 6$, $k = 3$, and the matrix `m` is the following one

```
0 1 0 1 0 1
1 0 1 0 1 0
0 1 0 0 0 1
1 0 1 0 1 0
0 1 0 0 0 0
3 0 1 0 2 0
```

the submatrix of size three with the largest sum is

```
1 0 1
0 1 0
3 0 1
```

and the one with the smallest sum is

```
0 0 1
0 1 0
0 0 0
```

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 includes an undefined number of rows. Each row includes two fields: A string of at most 100 characters, and an integer value. Those fields are separated by one or more spaces.
Write a C function to organize the file content into a list of lists. The elements of the main list represent different strings (i.e., strings are used as the main key of the data structure). The elements of the secondary lists represent different integers (i.e., integers are adopted as the secondary key of the data structure). More.texifically:

- - All records of the file with the same string must be stored in the same secondary list.
- All records of the file with the same string and the same integer value must be stored in the same element of a secondary list. Each element of the secondary list also stores the number of rows with the same string and integer in the file.

Write the function:

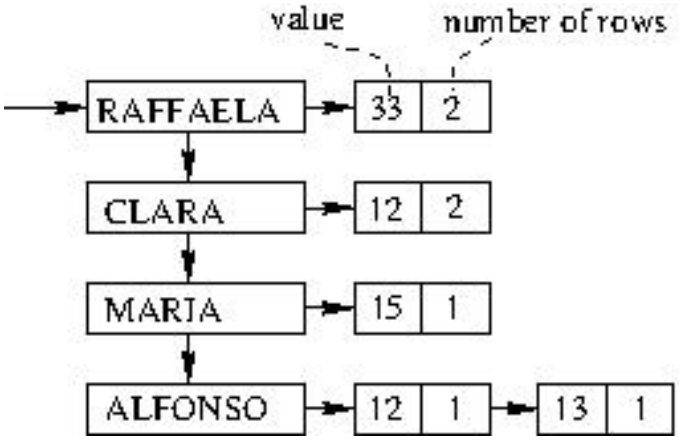
```
list1_t insert (char *name);
```

which receives the file name and returns the pointer of such a list.
Define the type list1_t and list2_t for the primary and secondary lists. Lists do not have to be sorted. The string within the list1_t type must be dynamically allocated.
For example, if we suppose that the file has the following content

```
ALFONSO 13
MARIA 15
CLARA 12
RAFFAELA 33
```

```
ALFONSO 12
CLARA 12
RAFFAELA 33
CLARA 12
```

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)
A basic calculator supports the four basic operations on integer decimal numbers: Division, multiplication, addition, and subtraction. All operations can be applied only to integer numbers and return an integer number (including the division, i.e., $11/5 = 2$). Moreover, all operators have the same precedence and they are left-associative. For example, the expression $22 - 21 * 7/3$ must be evaluated as $((22 - 21) * 7)/3 = 2$.
Write the function

```
void calculator (int *v, int n, int result);
```

which receives an array v of size n of integer values, and the integer value result, and it displays all possible expressions delivering the result.
For example, if `v = {3, 2, 25, 5}`, that includes four integer values (i.e., `n=4`), and the result of the expression is 5 (i.e., `result=5`), the expressions leading to that value (and that the function must display) are $(2 * 3 * 5 - 25)$, $(3 * 2 * 5 - 25)$, $(3 * 5 * 2 - 25)$, etc., $(25/3 - 5 + 2)$, $(25/3 + 2 - 5)$, etc.