University of M'sila
Faculty of Mathematics and Computer Science
Department of Computer Science

جامعة المسيلة
كلية الرياضيات و الإعلام الآلي
قسم الإعلام الآلي

$Exam : Data Structures and Algorithms 3 (DSA 3)$

Duration: 2h - Documents: not allowed **Academic year : 2025 / 2026**

Date : 17/01/2026 (10 : 30 – 12 : 00) **Level : L2 Semester : 5**

First Name	Last Name	Group
.....	.....	.....

Exercise No.	1	2	3	4	5	6	Total
Marks Obtained							

Exercise 1 : (1.5 Points / 10 Minutes)

Write the values of the **int** variables when the following program ends.

```
#include <stdio.h>
void main() {
   int a[] = {4, 5, 8, 13, 20};
   int *p, *q;
   int i, j, k, l, m, n;
   q = &a[4];
   p = q - 4;
   i = *q;
   j = *p++;
   k = *--q;
   l = p[1];
   m = *(q-2);
   n = q - p;
}
```

Answer

i = ...

j = ...

k = ...

l = ...

m = ...

n = ...

Exercise 2 : (5 Points / 30 Minutes)

Q1) Prove that $\mathbf{T(n) = n^2 + \log(n^2)}$ is $\mathbf{O(n^2)}$ using the formal definition of the Big-Oh notation. **Hint**: Find a constant c and threshold n_0 such that $\mathbf{T(n) \leq cn^2}$ for $\mathbf{n \geq n_0}$.

.....

.....

.....

.....

.....

Q2) Calculate the complexity of the following sources fragments; briefly justify your answer:

Pieces of code	Complexity

`int sum = 0;`
`for(int i = n*n*n*n; i >= 1; i /= 4)`
`  sum++;`
	……
	……
	……
	……
	……

`for(int i = 1; i < size; i++){`
`  current = arr[i];`
`  for(j = i; j>0 && arr[j-1]>current]; j--)`
`  arr[j] = arr[j-1];`
`  arr[j] = current;`
`}`
	……
	……
	……
	……
	……
	……
	……

`int sum = 0;`
`for (int i = 0; i < n/2; i++){`
`  for (int j = 0; j < n; j++)`
`  sum++;`
`  for (int j = 1; j < n * n; j++)`
`  for(int k = 1; k < 2026; k++)`
`  sum++;`
`}`
	……
	……
	……
	……
	……
	……
	……
	……

`int func(int n){`
`  for (int i = 0; i < n; i ++)`
`  // some operations`
`  if (n <= 0)`
`  return 1;`
`  else`
`  return 1 + func(n-1) + func(n-1);`
`}`
	……
	……
	……
	……
	……
	……
	……
	……

`int count = 0;`
`for (int i = n; i > 0; i /= 2)`
`  for (int j = 0; j < i; j++)`
`  count++;`
	……
	……
	……
	……
	……

Q3) You must find **1000** most expensive items from an unsorted price list containing $\mathbf{10^7}$ different items. Two schemes of solution are as follows.

- **Scheme A**: repeat **1000 times** the sequential search (with the linear complexity $\boldsymbol{O(n)}$).
- **Scheme B**: convert the list into an array (the same complexity $\boldsymbol{O(n)}$ as for search), then sort the array (complexity $\boldsymbol{O(n \log n)}$) and fetch **1000** top items.

Which scheme would you prefer assuming that searching for **100 items** in the unsorted list takes **0.1 millisecond (ms)** and sorting of **100 items** also takes **0.1 ms**? **Time for fetching data should not be taken into account**.

……

……

……

……

……

……

……

……

……

……

Exercise 3 : (5 Points / 25 Minutes)

The stooge sort is a recursive sorting algorithm that can be written quite simply:

```
void stooge_sort(int Tab[], int start, int end) {
    if (Tab[start] > Tab[end]) {
        swap(Tab[start], Tab[end]);
    }
    if (end - start + 1 > 2) {
        int t = (end - start + 1) / 3;
        stooge_sort(Tab, start, end - t);
        stooge_sort(Tab, start + t, end);
        stooge_sort(Tab, start, end - t);
    }
}
```

Q1) Run the algorithm for `stooge_sort([8, 9, 2, 7], 0, 3)`. During the execution, you must rewrite the contents of the array **`Tab`** each time values are swapped.

For each new recursive call, you must indicate the values of the **`start`** and **`end`** indices.

Call `stooge_sort(Tab, start, end)`	**Tab are swapped**
`stooge_sort([8, 9, 2, 7], 0, 3)`	**`[7, 9, 2, 8]`**
……	……
……	……
……	……
……	……
……	……
……	……
……	……
……	……
……	……
……	……
……	……
……	……

Q2) Give the recurrence equation for the number of recursive calls performed by **`stooge_sort(Tab, 0, n – 1)`**, where **`n`** is the number of elements in **`Tab`**.

……

……

……

Q3) Deduce the asymptotic complexity of sorting from the recurrence equation.

……

……

……

……

……

Q4) What should we think about the complexity of this sorting method compared to Bubble sort and Quick sort?

……

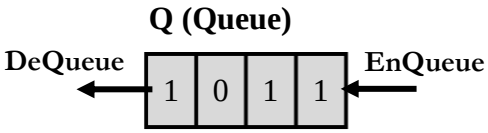
……

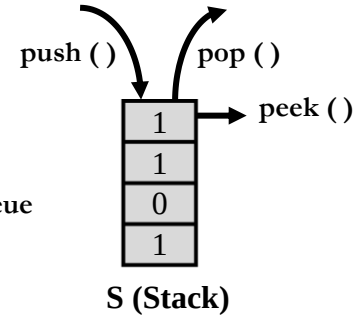
……

Exercise 4 : (4 Points / 25 Minutes)

In the example below, we have a **queue Q** and a **stack S**, each containing the binary representation of the integer number **13** (**1101**).

13	**2**
1	**6**
0	**3**
1	**1**
1	**0**





Q1) Provide the declaration for this queue and this stack.

Node	Queue	Stack
.....	.....	.....
.....	.....	.....
.....	.....	.....
.....	.....	.....
.....	.....	.....
.....	.....	.....

In all questions, you can use all functions seen in course without needing to redefine them.

Q2) Write the function **`binary(x, Q)`** which receives a decimal number **`x`** and creates a queue **`Q`** containing its binary representation.

```
void binary(int x, QUEUE *Q) {
```

……

……

……

……

……

……

……

```
}
```

Q3) Write a **`Stack_Reverse(S)`** that destructively returns a stack with the same elements as its input stack but in reverse order.

```
STACK Stack_Reverse (STACK S){
```

……...

……...

……...

……...

……...

……...

```
}
```

Q4) Write the function **`egual(Q, S)`** that makes it possible to verify whether the queue **`Q`** and the stack **`S`** contain binary representations of the same integer number.

```
bool egual(QUEUE Q, STACK S) {
```

……...

……...

……...

……...

……...

……...

……...

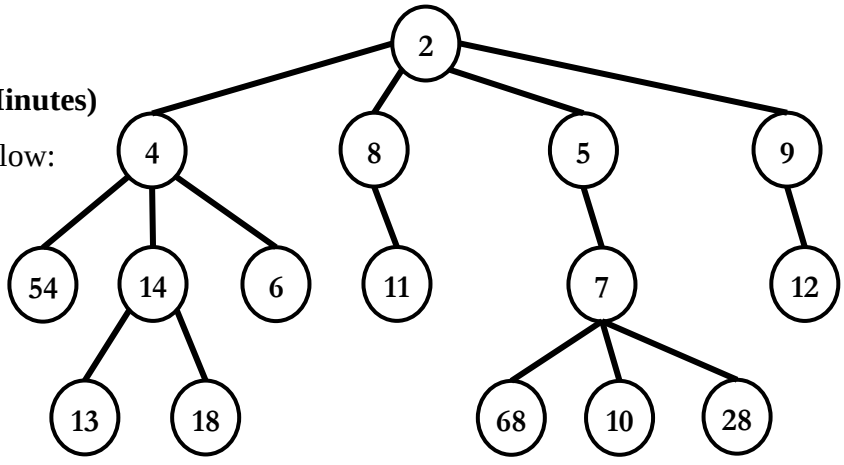
……...

……...

```
}
```

<u>Exercise 5 :</u> (4.5 Points / 20 Minutes)

For the general tree ***T*** shown below:



Q1) Find the corresponding binary tree T'.

……

……

……

……

……

……

……

……

……

……

……

Q2) In the binary tree T', write the pre-order, in-order and post-order traversals of each one.

Pre-order	…… ……
In-order	…… ……
Post-order	…… ……

Q3) Is the T' a binary search tree? Why?

……

……

……

A node in a binary tree is defined as follows:

```
struct BTNode {
    int data;
    BTNode *left;
    BTNode *right;
};
typedef BTNode *BTREE;
```

Q4) Write a **`Height(BTree)`** function to find a height of any node in the binary tree.

```
int Height(BTREE BTree) {
```

……

……

……

……

……

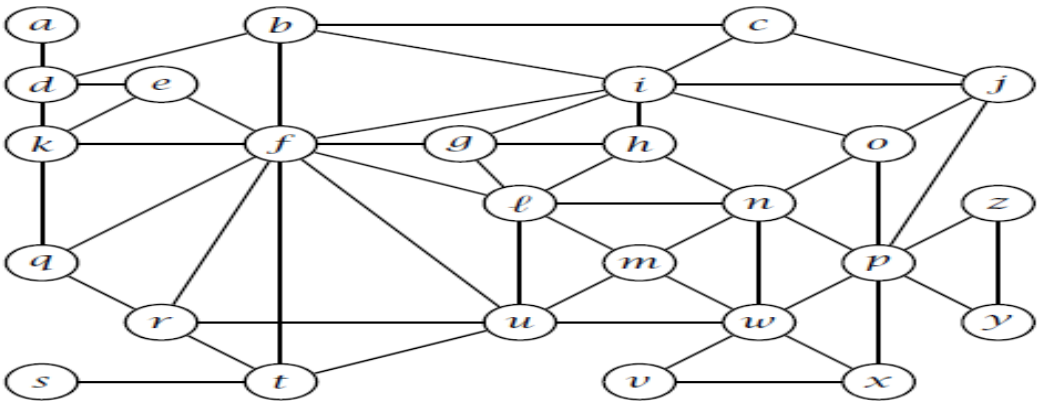
……

……

}

Exercise 6 (Bonus): (2 Points / 10 Minutes)

Consider the following graph. Suppose we want to traverse it, starting at node ***l***.



Q1) If we traverse this using **Breadth-First Search (BFS)**, what is a possible order of the nodes we visit? When processing the neighbors of a vertex, process them in alphabetical order.

……

……

Q2) Write the structure of the graph to represent it using the adjacency list.

……

……

……

……

……

……

Good Luck