Homework n°3

Computer Systems Organization

1 Truth table to diagram

Consider the logical function defined by the following truth table:

a	b	$F(a,b)$
0	0	1
0	1	0
1	0	1
1	1	1

Provide the diagram of this function using only NAND gates.

2 Linked list

Define a linked list storing integers (you can use a struct to define a type "node" with the fields val and next).

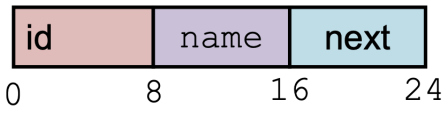
1. In the `main` function, implement by hand the first three nodes storing the values 5, 3, 2.
2. Write a `while` loop to print the values of the linked list you just made.
3. Write a `find` function that takes as parameters a pointer to a node and the integer to find, and returns a pointer to the node containing the desired value.

2.1 Binary puzzle

Let us create a structure in C that represents a node in a linked list:

```
1 typedef struct node {
2   long id;
3   char *name;
4   struct node *next;
5 } node;
```

A graphical representation:



Suppose that the register %rdi contains n.

2.1.1 Question 1:

What would be the equivalent in x86-64 Assembly of the following lines in C:

```
1 n->id = 10;
2 n->name = NULL;
3 n->next = n;
```

What do you have to add in your main to make this code work?

2.1.2 Question 2:

You are given the following snippet in Assembly:

```
  jmp .L1
.L3:
  cmpq    %rsi, (%rdi)
  jne     .L2
  movq    8(%rdi), %rax
  ret
.L2:
  movq    16(%rdi), %rdi
.L1:
  testq   %rdi, %rdi
  jne     .L3
  movq    $0, %rax
  ret
```

Based on this snippet, and knowing that:
%rdi has the value of n
%rsi has the value of id
%rax should contain the return value,
fill in the blanks in the following C code:

```
?? mystery(node *n, long id) {
   ???
}
```

3 Malloc

Complete the missing parts of the following program in C (use `malloc` in the `insert_front` function):

```c
#include <stdio.h>
#include <stdlib.h> // For malloc

typedef struct node {
    int val;
    struct node *next;
} node;

// insert val in the front of the linked list
// returns new head
void insert_front(node **headdp, int val) {
   //
   // WRITE YOUR CODE HERE
   //
   //
}

int main() {
    node *headp = NULL; // Initially, the list is empty
    // Insert elements at the front of the list
    for (int i = 0; i < 3; i++) {
        insert_front(&headp, i);
    }

    // Print the list to verify
   //
   // WRITE YOUR CODE HERE
   //
   //

     // Free the allocated memory
```

```
32      //
33      // WRITE YOUR CODE HERE
34      //
35      //
36
37
38       return 0;
39  }
```