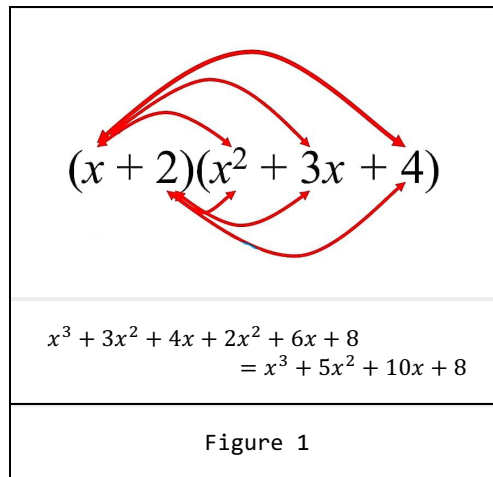


Exercise I: Lab Question (25 points)

Consider the Polynomial lab session. Write the function *multiply*, which given two polynomials, returns a third polynomial representing their multiplication. You should use the exact data types used in lab. All polynomials are sorted according to the decreasing order of the exponents. Check figure 1 to recall polynomial multiplication.



$$(x + 2)(x^2 + 3x + 4)$$

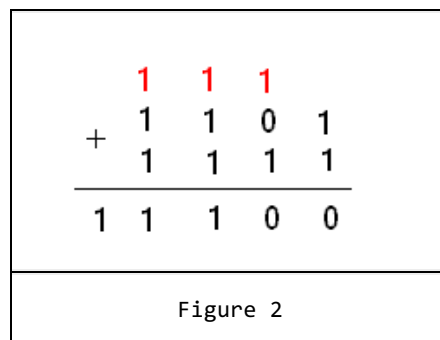
$$x^3 + 3x^2 + 4x + 2x^2 + 6x + 8$$

$$= x^3 + 5x^2 + 10x + 8$$

Figure 1

Exercise II: Two Big Binary Numbers Addition (25 points)

Write a function *add* which, given two text file names, where each file contains a huge binary number, represented as a series of '0' and '1' characters, adds the binary numbers and writes the result in a third text file whose name is the concatenation of both file names, separated by the word ADD. For example, if the file "file1.txt" contains 1101 and the file "file2.txt" contains 1111, then after calling the function *add* over the two file names, we should obtain the file "file1ADDfile2.txt" containing 11100.



$$\begin{array}{r}
 1101 \\
 + 1111 \\
 \hline
 11100
 \end{array}$$

Figure 2

Exercise III: Flattening a Linked List (28 points)

Given a linked list where every node represents a linked list and contains three pointers of its type.

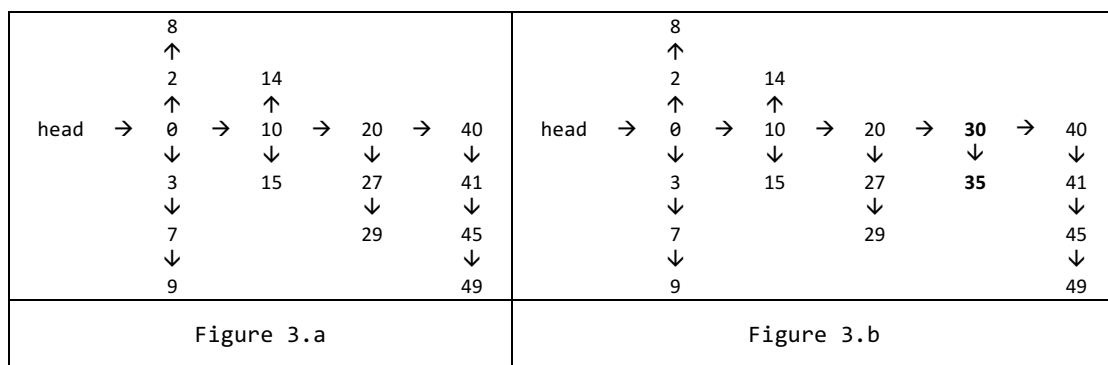
```
typedef struct node{
    int data;
    struct node *next, *upEven, *downOdd;
} node;
```

The main list nodes :

- their data values are rounded decimals to tenth,
- their upEven field points to the even values list belonging to the same range of tenth, and
- the downOdd field points to the odd values list belonging to the same range of tenth.

The nodes in the even lists have always their next and downOdd fields set to NULL. Similarly, the nodes in the odd lists have always their next and upEven fields set to NULL. All linked lists are sorted.

See the example in figure 3.a.



1. Write the function **put** which takes an integer and pushes it to its correct place in this data structure. Note that, if the node representing the rounded decimal to the tenth of the given integer is missing, the function should first add the rounded decimal to the structure, then add the node corresponding to the given integer. Example, if we put 35 into the list in figure 3.a, we obtain the list in figure 3.b.
2. Write the function **flatten** which flattens the data structure into a sorted linked list. The provided solution should not use any dynamic allocation or deallocation. Moreover, while flattening, each node should be visited only once. For example, for the above input list in figure 3.a, output list should be:

0→2→3→7→8→9→10→14→15→20→27→29→40→41→45→49