Part A: Recursion

Exercise 1

Given this recursive function:

```
int Foo (int v[], int n)
{
     if (n == 0)
          return 0;
     else
          return Foo (v,n-1)*10+v[n-1];
}
```

1. Indicate in the table the sequence of calls that are made until the result is obtained, the parameters and the result of each recursive call. The first line of the table corresponds to the first recursive call to the function.

2. Implement an iterative function that does the same thing as this recursive function.

3. How many recursive calls are made to find the solution? How many iterations of the loop in the iterative version? Which solution seems more efficient to you? Why?

Call	v	n	Result
1	[3, 5, 7, 4, 6]	5	

DELIVERY: pdf file with the answers.

Part B: Hashing

Exercise 1

Considering the following resultant hash tables:

0	
1	
2	2
3	23
4	
5	15
6	
7	
8	18
9	

(A)

0	
1	
2	12
3	13
4	
5	5
6	
7	
8	18
9	

(B)

0	
1	
2	12
3	13
4	2
5	3
6	23
7	5
8	18
9	15

(C)

0	
1	
2	12, 2
3	13, 3, 23
4	
5	5, 15
6	
7	
8	18
9	

(D)

A) The keys 12, 18, 13, 2, 3, 23, 5 and 15 are inserted into an initially empty hash table of length 10 using open addressing with hash function h(k) = k mod 10 and linear probing. Which of the hash tables presented above is the resultant hash table?

B) Which of the following choices give a possible sequence in which the key values might have been inserted in table D? Which approach to resolve collisions has been applied?
(A) 12, 2, 13, 3, 23, 5, 15, 18
(B) 12, 18, 13, 2, 3, 23, 5, 15
(C) 12, 13, 5, 18, 2, 3, 23, 15
(D) 13, 5, 15, 18, 12, 2, 3, 23

DELIVERY: pdf file with the answers.

Exercise 2

Assume the following data structure as a hash table:

```
struct Element{
   int data;
   int key;
};
```

And the hash function h(key)=key mod 10 with linear probing.

Given the provided template `hash.c`, implement the following functions of the hash table:

- Insert a new key-value pair (a new Element) to the hash table

```
void insert(int key, int data);
```

- Remove a given element from the hash table

```
void removeElement(struct Element * element);
```

DELIVERY: source file .c with the implemented functions.