Tutorial 6 – Exercises on Lists

The following exercises modify a linked list of integers defined by the following typedef:

```
typedef struct elem {
    int data;
    struct elem* next;
} elem;
```

Each solution should be tested with random lists whose sizes are between 0 and 10 elements. The function *rand()* in stdlib.h can be used to generate random numbers. The random seed should be set using *srand(time(NULL))*. The numbers should be inserted using the add_head() function seen in the lecture.

Note that when the input list has to be modified, a reference to the head point is given (not the head pointer itself). This means that a double reference to the list is provided:

```
elem** phead;
```

1. Remove the tail of a list passed as parameter. A free() call should be made on the element to remove:

   Prototype: `void remove_tail(elem** plist);`

2. Given a list, return a pointer to its n-th element (an integer n is passed is as parameter), where n=1 is the first element. If the integer is larger than the size of the list, return a NULL pointer. The element should not be removed from the list.

   Prototype: `elem* getelem(int n, elem* plist);`

3. Given an array of integers as input, create a list with the same elements allocating memory for the struct elem, and return the head pointer to the new list:

   Prototype: `elem* make_list(int* array, int len);`

4. Given a list, create two lists with the same number of elements that split in half the original list. The pointers for the front and back list should be set by the function. The same elements from the original list should be used. If the lists have an odd number of elements, the back list has an element more than the first.

   Example: input 0 3 4 2 1, output: front: 0 3 back: 4 2 1

   Prototype: `void split(elem* head, elem** front, elem** back);`