## Exercise 7a. (More operation functions for dynamically linked list, 1p)

In exercise 6 you implemented some operation functions for dynamically allocated linked list. The type definition of LinkedList was

```
template <class T>
class LinkedList {
private:
     // definition of the list node class
     class Node {
          …
     };

     Node *first;    // no *last variable
                     // in this implementation!!
     …
};
```

Continue to develop further the program of exercise 6. Write the following operation functions for a character list:

```
void delete_first();
void delete_last();
bool find_pos(T item, int &pos);
```

The find_pos function returns true if item is found and false otherwise. If the item is found the order number of it is passed to the caller in the parameter pos.

Use the following main function to test your new list operations:

```
int main() {
    LinkedList<char> list;
    int              order_no;
    char             to_be_searched;

    list.delete_last();

    list.insert_to_end('?');
    list.delete_last();

    list.insert_to_end('x');
    list.insert_to_end('a');
    list.insert_to_end('b');
    list.insert_to_end('c');
    list.insert_to_end('d');
    list.insert_to_end('y');
    list.print();

    cout << "Enter first character to be searced\n";
    cin >> to_be_searched;
    if (list.find_pos(to_be_searched, order_no))
        cout << "The order no is " << order_no << endl;
```

```
        else
            cout << "Not found\n";

        cout << "Enter second character to be searced\n";
        cin >> to_be_searched;
        if (list.find_pos(to_be_searched, order_no))
            cout << "The order no is " << order_no << endl;
        else
            cout << "Not found\n";
        list.delete_first();
        cout << list;

        list.delete_last();
        cout << list;

        return (EXIT_SUCCESS);
    }
```

## Exercise 7b. (Extra exercise, Ordered dynamically linked list, 0.25p)

In the exercise 3b we practiced with an ordered list. That ordered list was implemented using a simple array implementation.

Implement now that ordered list using linked list structure. Add the `list.insert(item)` function to the lab7a program. This operation function insert an item to the list in such a way that the list is always ordered (smallest item first).

Use the same test program than in lab3b, this demonstrates again benefits of abstraction and generality paradigm. We can modify the internal operation of the list totally, and there is no need to modify the user (main program) of the list at all.