**Exercise 1**

1. **void Insert**: using the method findFirstOccurrance (implemented in doubly-linked-linear-list.h) we check to see if there are any other occurrences of the inserted number. If the number already exists, a proper message will be displayed on the screen, otherwise the number will be added on the list using the addition operation;
2. **void sortedListInsert**: in order to insert an element and keep the list sorted, the program uses a while: while the node is not null and the value from the node is less than the added value,p (an auxiliary node declared previously) gets the next value. If the node is null,then we add the value of x, else we create the node which gets the value of x. Its next is going to be p and its previous p->prev. Paux is actually going to be the next of p previous. If this doesn’t happen,paux is p first and the previous of p is also paux.
3. **int dividedByZNr:** I created a new node in which we save the first node. As long as it is not null,the “if” command is executed. The “if” compares the value of paux with the arbitrary value chosen for z and if the rest is 0,then the number of numbers divisible by z is going to be increased. Paux takes the next value and the procedure starts again;
4. **int greaterThanFirstNr:** the rationament is pretty much the same, the only difference is that this time we compare the first value with the other values of the nodes in the list;
5. **int findOccurenceNr:** we copy the first node in an auxiliar node. As long as the value from a node is equal to the value given to the program,the number of occurrences will increase.

**Exercise 2**

1. **class Point** is where I defined hr actual point. It has two methods that overrides the << and < operators so that the points list is displayed using cout << ,respectively to be able to compare the distance between two points

**void ex2:** firstly, a new list of points is created, then using specific messagesthe program gets the values for x and y for as many points as the person using the program wants. A new object (point) is created. This object is in fact a point of coordinates x and y, a point which is going to be added in the list.

1. **void sortAscending:** this is a bubble sorting algorithm . I crated a new node where I copy the first node from the list. While this node is not null, a second auxiliary node is created ,this one being the node next to the first one. While the second node crated is not null ,the values of those two node are compared and if the second one is bigged than the first one, their place is changed.

**Exercise 3**

1. **A[i][j] == A[j][i]** for undirected graphs, so getting rid of the symmetry is what gave us the oriented graph
2. **Void ex3:** the nodes are the edges of the graph,so we make sure we get the values carefully guided by some explicit messages