POO LAB 4

Write the template **SingleLinkedList** (modeling the behavior of a single linked list) so that the following code:

```
void print(const int& value) { printf("%d ", value); }
int main() {
    SingleLinkedList<int> s;
    (((s += 10) += 20) += 30);
    s.print(print);
    if (s && 20) {
        printf("\n 20 is in the list !");
    }
    if (!(s && 25)) {
        printf("\n 25 is not in the list !");
    }
    while (s) {
        printf("\n%d", s.pop_first());
    }
}
```

compiles and upon execution prints the following to the screen:

```
10 20 30

 20 is in the list !
 25 is not in the list !
10
20
30
```

Observations:

- You are not allowed to use **STL** at all (for vectors, strings, maps or any **template/object** defined in STL). The only exception is the usage of "**std::cout**" from the main function
- Make sure that your template will work for both constants (e.g. number) as well as for a generic type
- We will only offer points for a single linked list based on a template (any other implementation will not be graded).

Grading (informative):

G1	Constructor for SingleLinkedList	1p
G2	Destructor for SingleLinkedList	2p
G3	Organize your project in 2 files: **main.cpp**, **SingleLinkedList.h**	1p
G4	Organize your template SingleLinkedList to include private and public members, the definition of a constructor and destructor, and at least one method.	2p
G5	Operator += (that ads a value to that links and returns a self reference)	6p
G6	Method: print that receives a pointer to a function that is called for every value in the list	4p
G7	Operator && that checks if a value exists in the list	4p
G8	Method: pop_first() that returns the first element in the list	4p
G9	A cast operator to bool that returns true if the list contains at least one element.	1p
G10	A template for a structure Node define in the same header as the class Single Linked List	2p
G11	Program compiles and upon execution produces the expected results	3p