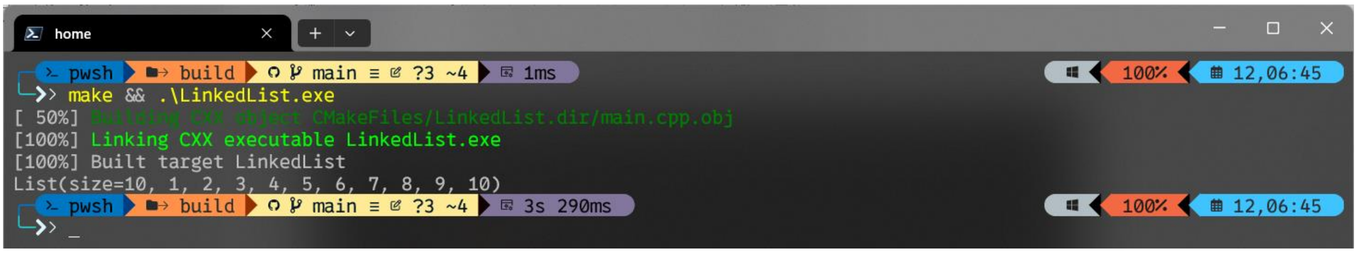
Implement the following tasks for the linear linked list:

1. Write a program that creates a list, inserts the integers 1 through 10, and then iterates through the list twice, printing its contents.

```cpp
#include <iostream>
using namespace std;

#include "LinkedList.hpp"

// main
int main(int argc, char const *argv[]){
    // Initialize list
    LinkedList<int> list;

    // insertion
    for(int x = 1; x <= 10; x++){
        list.push_back(x);
    }

    // display
    list.display();

    return EXIT_SUCCESS;
}
```

We will need this in next sections

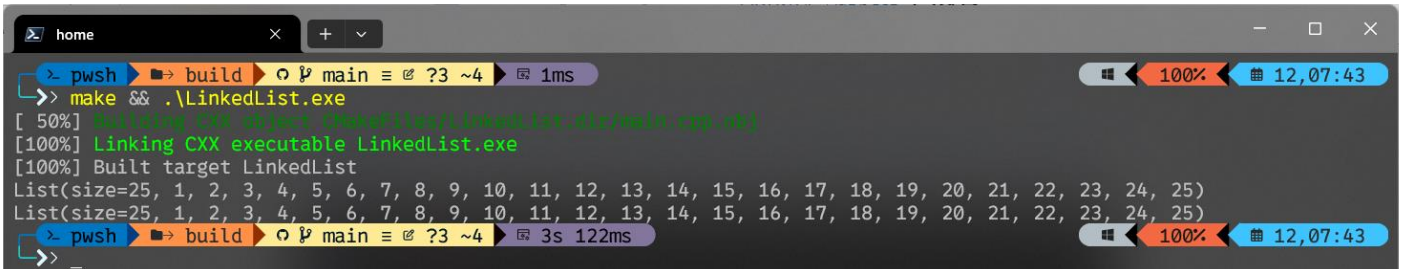
```cpp
template<class Type>
void LinkedList<Type>::display() const{
    cout << toString() << endl;
}

template<class Type>
string LinkedList<Type>::toString() const{
    string result = fmt::format("List(size={}", _length);

    for(Node* it = _head; it != nullptr; it = it->next){
        result += fmt::format(", {}", it->value);
    }

    result += ")";
    return result;
}
```



```
home
pwsh  build  main ≡ ?3 ~4  1ms                                         100%  12,06:45
>> make && .\LinkedList.exe
[ 50%] Building CXX object CMakeFiles/LinkedList.dir/main.cpp.obj
[100%] Linking CXX executable LinkedList.exe
[100%] Built target LinkedList
List(size=10, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
pwsh  build  main ≡ ?3 ~4  3s 290ms                                    100%  12,06:45
>> _
```

22. Write a program that creates two list L1 and L2, puts the integers 1 through 25 into L1, iterates through list L1 and copies its contents into list L2, and then iterates through list L2 and prints its contents.

```cpp
#include <iostream>
using namespace std;
#include "LinkedList.hpp"


// main
int main(int argc, char const *argv[]){
    // Initialize list
    LinkedList<int> list1, list2;

    // insertion
    for(int x = 1; x <= 25; x++){
        list1.push_back(x);
    }

    /** Hint
     * Instead of make this manually
     *   we have created operator overloading[=] to make this.
     */
    list2 = list1;

    // display
    list1.display();
    list2.display();

    return EXIT_SUCCESS;
}
```



```
home
pwsh  build  main ≡ ?3 ~4  1ms                                         100%  12,07:43
>> make && .\LinkedList.exe
[ 50%] Building CXX object CMakeFiles/LinkedList.dir/main.cpp.obj
[100%] Linking CXX executable LinkedList.exe
[100%] Built target LinkedList
List(size=25, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25)
List(size=25, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25)
pwsh  build  main ≡ ?3 ~4  3s 122ms                                    100%  12,07:43
>> _
```

23. **Write a C++ function as a member of the list class to insert an item after specific item in the list and if not exist then print at the end of the list.**

```
template<class Type>
LinkedList<Type>& LinkedList<Type>::push_at(Node<Type>* node, const Type& value){
    Node<Type>* newNode = new Node<Type>(value);
    ASSERT(newNode, "Stack overflow!");

    if(node == nullptr) push_back(newNode);
    else{
        newNode->next = node->next
        node->next = newNode;
    }

    return *this;
}
```

37. **Write a C++ function as a member of the list class to insert an item in an ordered list.**

38. **Write a C++ function as a member of the list class to delete an item from a list.**

```
template<class Type>
LinkedList<Type>& LinkedList<Type>::pop_front(){
    Node<Type>* deletedNode = _head;
    _head = _head->next;
    delete deletedNode;
    return *this;
}
```

<u>Implement the following tasks for the linked list:</u>

1.Write isEmpty() function to check for empty list, and makeEmpty() function to make the linked list empty.

```

template<class Type>
bool LinkedList<Type>::isEmpty() const{
    return _head == nullptr;
}
template<class Type>
bool LinkedList<Type>::isNotEmpty() const{
    return !isEmpty();
}
```

```
template<class Type>
LinkedList<Type>& LinkedList<Type>::clear(){
    _head = _tail = 0;
    _length = 0;
    return *this;
}
```

- I know that is isn’t the best practice but it is so bad, so I want to know how to deallocate every node from memory.

2.Write a linear search algorithm for linked list.

```
template<class Type>
bool LinkedList<Type>::linear_search(const Type& value){
    for(Node<Type>* it = _head; it != nullptr; it = it->next) {
        if(it->value == value) return true;
    }
    return false;
}
```

3.Write a function to copy one linked list to another.

```
template<class Type>
LinkedList<Type>& LinkedList<Type>::operator=(const LinkedList<Type>& list){
    clear();

    for(Node<Type>* it = list._head; it != nullptr; it = it->next){
        push_back(it->value);
    }

    return *this;
}
```

- **As I don’t have time, I will make the second part of assignment in the next week.**