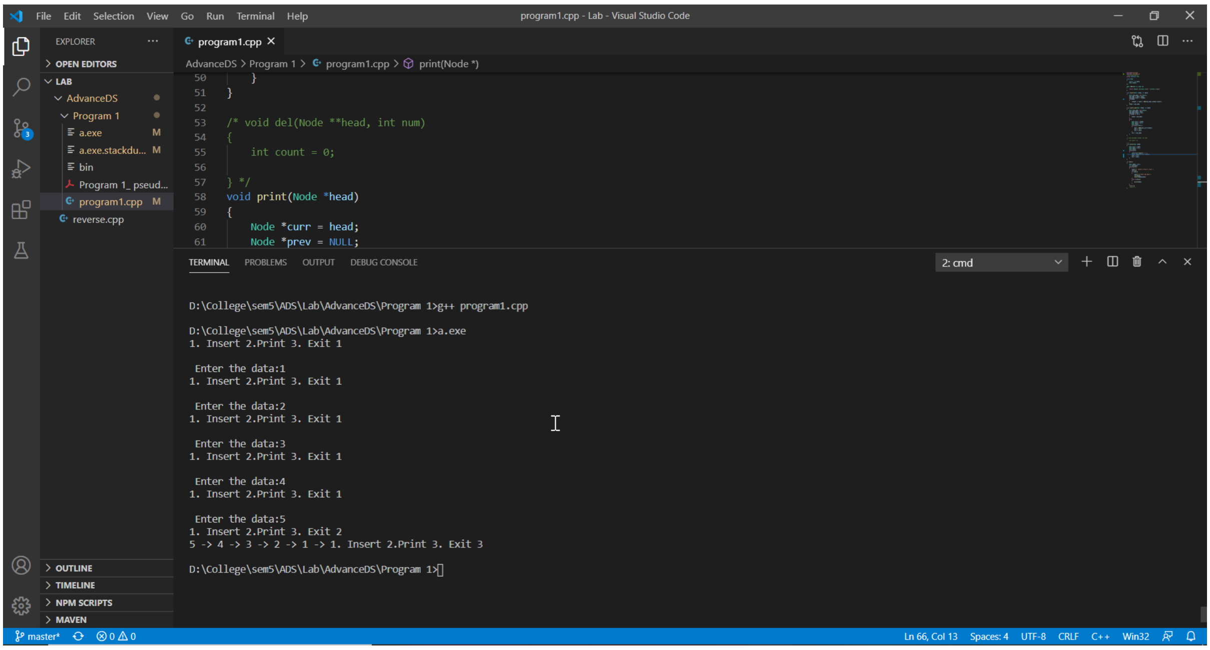
XOR linked list

Date	@Sep 23, 2020 12:53 PM
Files	
Tags	

Program 1:
Write a program to implement the following list: An ordinary Doubly Linked List requires space for two address fields to store the addresses of previous and next nodes. A memory efficient version of Doubly Linked List can be created using only one space for address field with every node. This memory efficient Doubly Linked List is called XOR Linked List or Memory Efficient as the list uses bitwise XOR operation to save space for one address. In the XOR linked list, instead of storing actual memory addresses, every node stores the XOR of addresses of previous and next nodes.
Implement a function to insert a new node at the beginning and at the end of the list.

OUTPUT:

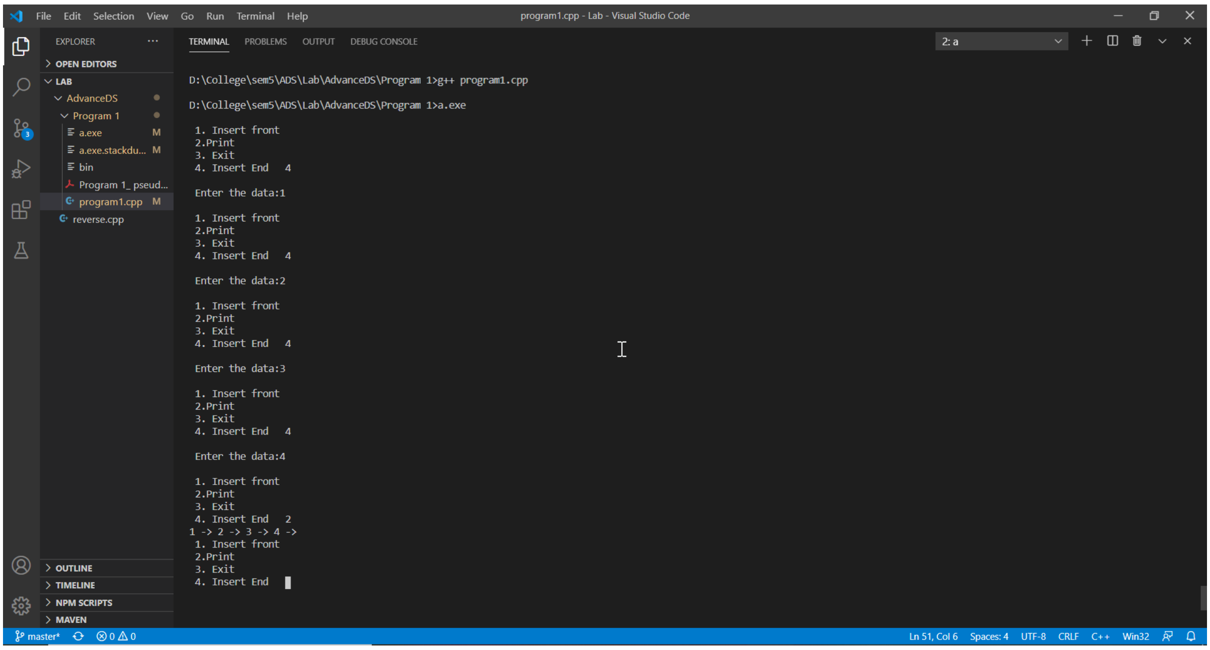
Insert front



```cpp
    }
}

/* void del(Node **head, int num)
{
    int count = 0;

} */
void print(Node *head)
{
    Node *curr = head;
    Node *prev = NULL;
```

```
D:\College\sem5\ADS\Lab\AdvanceDS\Program 1>g++ program1.cpp

D:\College\sem5\ADS\Lab\AdvanceDS\Program 1>a.exe
1. Insert 2.Print 3. Exit 1

 Enter the data:1
1. Insert 2.Print 3. Exit 1

 Enter the data:2
1. Insert 2.Print 3. Exit 1

 Enter the data:3
1. Insert 2.Print 3. Exit 1

 Enter the data:4
1. Insert 2.Print 3. Exit 1

 Enter the data:5
1. Insert 2.Print 3. Exit 2
5 -> 4 -> 3 -> 2 -> 1 -> 1. Insert 2.Print 3. Exit 3

D:\College\sem5\ADS\Lab\AdvanceDS\Program 1>
```

Insert end:



```
D:\College\sem5\ADS\Lab\AdvanceDS\Program 1>g++ program1.cpp

D:\College\sem5\ADS\Lab\AdvanceDS\Program 1>a.exe

 1. Insert front
 2.Print
 3. Exit
 4. Insert End   4

 Enter the data:1

 1. Insert front
 2.Print
 3. Exit
 4. Insert End   4

 Enter the data:2

 1. Insert front
 2.Print
 3. Exit
 4. Insert End   4

 Enter the data:3

 1. Insert front
 2.Print
 3. Exit
 4. Insert End   4

 Enter the data:4

 1. Insert front
 2.Print
 3. Exit
 4. Insert End   2
1 -> 2 -> 3 -> 4 ->
 1. Insert front
 2.Print
 3. Exit
 4. Insert End
```