Assignment 2

First name: Hima Sai Kiran
Last name: Prudhivi
NetID: HXP220011
Section: 001

First name: Anthea
Last name: Abreo
NetID: AXA210122
Section: 001

Question 1: Majority Element

Given an array nums of size n, return the majority element.

The majority element is the element that appears more than ⌊n / 2⌋ times. You may assume that the majority element always exists in the array.

Example 1:

Input: nums = [3,2,3]
Output: 3

Example 2:
Input: nums = [2,2,1,1,1,2,2]
Output: 2

Constraints:
n == nums.length
1 <= n <= 5 * 104
-109 <= nums[i] <= 109

Code

```cpp
/* Topic: Arrays
 * Question: Majority Element * Given an array nums of size n, return the majority element. * The majority element is the element that appears more than ⌊n / 2⌋ times. * You may assume that the majority element always exists in the array. */
#include <iostream>
#include<vector>

using namespace std;

int majorityElement(vector<int>& nums) {
```

```
    int ME;
    int me_count = 1;
    int n = nums.size();

    ME = nums[0];


    for(int i = 1; i < n; i++){

        if(me_count == 0){
            ME = nums[i];
            me_count = 1;
        }
        else{
            if(nums[i] == ME){
                me_count++;
            }
            else{
                me_count--;
            }
        }

    }

    return ME;
}

int main() {
    vector<int> arr = { 1, 1, 2, 1, 3, 5, 1 };

    // Function calling
    cout<<"The majority element is "<<majorityElement(arr);
    return 0;
}
```

Output

Question 2: Longest Consecutive Sequence

Given an unsorted array of integers nums, return the length of the longest consecutive elements sequence.

You must write an algorithm that runs in O(n) time.

Example 1:

Input: nums = [100,4,200,1,3,2]
Output: 4

Explanation:
The longest consecutive elements sequence is [1, 2, 3, 4]. Therefore its length is 4.

Code

```
/* Topic: Hashing
 * Question: Longest Consecutive Sequence * Given an unsorted array of integers
nums, return the length of the longest consecutive elements sequence. * You must
write an algorithm that runs in O(n) time. */
#include <iostream>
#include <vector>
#include <unordered_set>

using namespace std;

int longestConsecutive(vector<int>& nums) {

    int maxcount = 0;
    int ele;

    if(nums.size() == 0){
        return 0;
    }

    int count = 0;
    unordered_set<int> hset;

    //Insert elements in hash set
    for(int i = 0; i < nums.size(); i++){
        hset.insert(nums[i]);
    }

    //Check for consecutive sequence
    for(int i = 0; i < nums.size(); i++){

        ele = nums[i];

        if(hset.find(ele - 1) == hset.end()){

            count = 1;
```

```
            while(hset.find(ele + 1) != hset.end()){
                count++;
                ele++;
            }

        }

        maxcount = max(maxcount, count);
    }

    return maxcount;
}

int main()
{
    vector<int> arr = { 1, 9, 3, 10, 4, 20, 2, 5, 11 };
    cout << "Length of the Longest contiguous subsequence "
            "is "         << longestConsecutive(arr);
    return 0;
}
```

Output

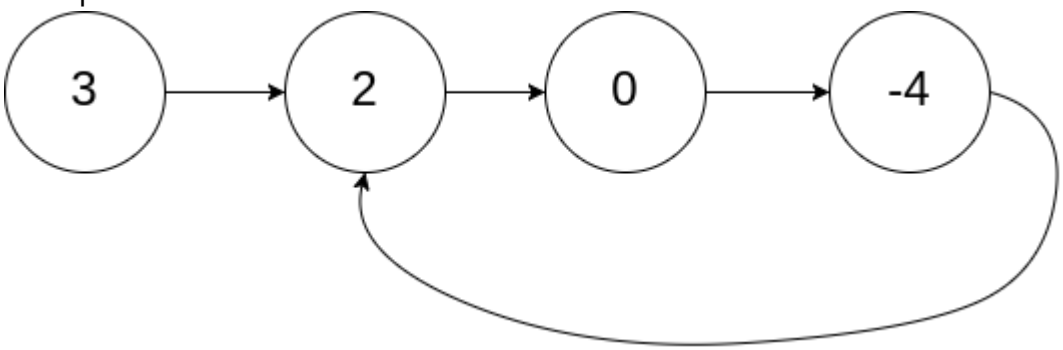
Question 3: Linked List Cycle

Given head, the head of a linked list, determine if the linked list has a cycle in it.

There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to. Note that pos is not passed as a parameter.

Return true if there is a cycle in the linked list. Otherwise, return false.

Example:



Input: head = [3,2,0,-4], pos = 1

Output: true
Explanation: There is a cycle in the linked list, where the tail connects to the 1st node (0-indexed).

Code

```
/* Topic: Linked Lists
 * Question: Linked List Cycle * Given head, the head of a linked list, determine if the linked list has a cycle in it. * Return true if there is a cycle in the linked list. Otherwise, return false. */#include <iostream>
#include <vector>

using namespace std;

struct ListNode {
    int val;
    ListNode *next;
    ListNode(int x) : val(x), next(NULL) {}
};

void push(struct ListNode** head_ref, int new_data)
{
    /* allocate node */
    struct ListNode* new_node = new ListNode(new_data);

    /* link the old list off the new node */
    new_node->next = (*head_ref);

    /* move the head to point to the new node */
    (*head_ref) = new_node;
}

bool hasCycle(ListNode *head) {

    ListNode* fast;
    ListNode* slow;

    fast = head;
    slow = head;

    while(fast != NULL && fast -> next != NULL){
        slow = slow -> next; //Move slow pointer by 1
        fast = fast -> next -> next; //Move fast pointer by 2

        if(slow == fast){
```

```
            return true;
        }

    }

    return false;
}

int main(){
    ListNode* head = new ListNode(4);

    push(&head, 20);
    push(&head, 5);
    push(&head, 15);
    push(&head, 10);

    head->next->next->next->next = head;


    if(hasCycle(head))
        cout<<"Loop found"<<endl;

    else
        cout<<"No Loop"<<endl;

    return 0;

}
```

Output