# CMPSC 24: Midterm #1
# WINTER 2017: January 31, 2016

NAME: ____________________

EMAIL: ____________________

Student ID: ____________________

| | | |
|---|---|---|
| Question 1 (20 points) | ___________ | [Time: 15 minutes] |
| Question 2 (20 points) | ___________ | [Time: 15 minutes] |
| Question 3 (20 points) | ___________ | [Time: 15 minutes] |
| Question 4 (20 points) | ___________ | [Time: 15 minutes] |
| Question 5 (20 points) | ___________ | [Time: 15 minutes] |
| **Total (100 points)** | ___________ | **[Time: 75 minutes]** |

**NAME: ____________________________ ID: ______________________________**

## Q1. [Array Palindrome]

Given an array of *integers*, write a function that returns *true* if the given array is palindrome, else *false*. A 'array palindrome' is an array which, when its elements are reversed, remains the same (i.e., the elements of the array are same when scanned forward or backward)

**Examples**:

a) Input array:
1, 2, 3, 4, 3, 2, 1
The input array is a palindrome

b) Input array:
1, 3, 5, 7, 9, 7, 5, 3, 2, 1
The input array is NOT a palindrome

```
bool palindrome(int input_array[],int size)

{
  for(int I = 0; I < size/2; i++)
  {
    if(input_array[i] != input_array[size-i-1])
          return false;
  }
  return true;
}
```

## Grader: Mohammad

**NAME: ___________________________ ID: _____________________________**

## Q2. [Reverse a Linked List]

The following code is a function for creating a copy of a given linked list. In this function we create a new empty linked list, start from the head of given link list (Source), traverse the linked list and copy the nodes one by one to the new linked list. Use this idea to write a function to reverse a linked list.

```
void IntLinkedList::copy(const IntLinkedList & Source)
{
  If (Source.Head == NULL){
    return;
  }
  Node *Current = Source.Head->Next;
  Node *Iterator = new Node(Source.Head->data, NULL);
  Head = Iterator;
  While (Current != NULL)
  {
    Iterator->next = new Node(Current->data, nullptr);
    Current = Current->Next;
    Iterator = Iterator->Next;
  }
}
```

```
void IntLinkedList::reverse(const IntLinkedList& Source)
{
  if (Source.Head == NULL){
    return;
  }
  Node *Current = Source.Head->Next;
  Node *Iterator = new Node(Source.Head->data, NULL);
  While (Current != NULL){
    Node *tmp = new Node(Current->data, Iterator);
    Iterator = tmp;
    Current = Current->Next;
  }
  Head = Iterator;
}
```

**Grader: Da**

**NAME: ________________________ ID: __________________________**

## Q3. [Palindrome Linked List]

Given a singly linked list of integers, write a function in pseudocode that returns *true* if the given list is palindrome, else *false*.
**[Hint: use copy and reverse functions from question 2]**

What is the time complexity of your function assuming that there are **N** elements on the list?

```
IntLinkedList*  copyList = new IntLinkedList();// Create a copyList
IntLinkedList* revList     = new IntLinkedList();// Create a revList

copyList = copyList->copy(input_list);// Copy inputList into copyList
revList   =  revList->reverse(input_list);// Set revList to reverse of input_list

Node* p1=copyList.Head; // Initialize p1 to head of the copyList
Node* p2=revList.Head; // Initialize p2 to head of the revList

while(p1!= NULL && p2!=NULL) //  while the nodes are not NULL
{
        If(p1->data != p2->data)   // if copyList and revList are not equal
                Return false;            //return false
        Else                             // else
        {
                p1=p1->next;           // traverse p1 to next node
                p2=p2->next;           // traverse p2 to next node
        }
}
Return true;                             //return true
```

**NOTE: Either code or pseudocode is accepted.**

**Time Complexity: O(n)**

**Grader: SHAMAN BHAT**

**NAME: ___________________________ ID: _____________________________**

## Q4. [Array Based Implementation of List]

Given below is the header file for a class called ListType_Q4.h defined as follows:

```
#include "ItemType.h" // type definition declared here
class ListType_Q4
{
 public:
  ListType_Q4();
  int DeleteFromFirstPosition();

 private:
   int length;
   int info[MAX];
 };
```

Your task is to provide the ListType_Q4.cpp implementation. Here is the skeleton of the ListType_Q4.cpp file (do not worry about the issue of list being full and list being empty):

```
// Implementation file for ListType_Q4.h
#include "ListType_Q4.h"
ListType_Q4::ListType_Q4()
{
        length = 0;

}
// This method deletes the first element at array position ZERO and returns // it as a return parameter and moves the remaining elements one position // up while maintaining the order of elements
int ListType_Q4::DeleteFromFirstPosition()
{
            int itemToReturn  = info[0];

            for (int i = 1; i < length; i++)
            {
                        info[i-1] = info[i];
            }

            length--;

            return itemToReturn;

}
```

- What is the time complexity of the *DeleteFromFirstPosition* method assuming that there are **N** elements on the list?

  **O(n)**

**NAME: ___________________________ ID: _____________________________**

## Q5. [Linked List Based Implementation of List]

Given below is the header file for a class called ListType_Q5.h defined as follows:

```
#include "ItemType.h"
class NodeType
{
 public:
    NodeType(const int& init_val = ItemType(),
                        NodeType* init_link = NULL)
    { info = init_val; next = init_link;}
    void set_info(const int& new_info) { info = new_info;}
    void set_link(NodeType* new_link) {next = new_link;}
    int get_info() const {return info;}
         NodeType* get_link() {return next;}
  private:
         int info;
         NodeType* next;
};

class ListType_Q5
{
 public:
  ListType_Q5();
  void InsertAtFirstPosition(const int& item);

 private:
         int length;
         NodeType* list;
 };
```

Your task is to provide the ListType_Q5.cpp implementation. Here is the skeleton of the ListType_Q5.cpp file (do not worry about the issue of the list being empty):

```
// Implementation file for ListType_Q5.h
#include "ListType_Q5.h"
ListType_Q5::ListType_Q5()
{
             length = 0;
             list = NULL;
}
```

**NAME: ___________________________ ID: _____________________________**

```
// This method takes the input parameter "item" and inserts it as the // first node in the linked list
void ListType_Q5::InsertAtFirstPosition(const int& item)
{

    list = new NodeType(item, list);
    length++;



}
```

- What is the time complexity of the *InsertAtFirstPosition* method assuming that there are **N** elements on the list?
  **O(1)**