Sardar Vallabhbhai National Institute of Technology, Surat
Department of Artificial Intelligence
Data Structure (AI102)
B.Tech I - II Semester

Assignment-4

Write C/C++ program for the following

Q1: Given a linked list of n nodes and an integer k, write a function to rotate the linked list counter clockwise by k nodes.

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct node {
    int data;
    struct node *next;
} node;

// Function to create a new node
node* createnode(int data) {
    node* newnode = (node*)malloc(sizeof(node));
    newnode->data = data;
    newnode->next = NULL;
    return newnode;
}

// Function to display linked list
void display(node* head) {
    node* temp = head;
    while (temp != NULL) {
        printf("%d->", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
```

```c
// Function to rotate linked list left by k positions
node* rotate_left(node* head, int n, int k) {
    if (head == NULL || k == 0) return head;

    node* temp = head;
    int count = 1;

    while (count < k && temp->next != NULL) {
        temp = temp->next;
        count++;
    }

    if (temp->next == NULL) return head;  // If k >= n, return original list

    node* new_head = temp->next;
    temp->next = NULL;  // Break the list at kth node

    node* tail = new_head;
    while (tail->next != NULL) {
        tail = tail->next;
    }
    tail->next = head;  // Connect the tail to the old head

    return new_head;
}

// Main function
int main() {
    int n, k, i, data;

    printf("Enter the number of nodes: ");
    scanf("%d", &n);

    // Dynamically allocate an array of node pointers
    node** nodeArray = (node**)malloc(n * sizeof(node*));

    printf("Enter data elements: ");
    for (i = 0; i < n; i++) {
        scanf("%d", &data);
        nodeArray[i] = createnode(data);
    }
```

```
    // Link the nodes
    for (i = 0; i < n - 1; i++)
        nodeArray[i]->next = nodeArray[i + 1];

    printf("Enter k (number of positions to rotate left): ");
    scanf("%d", &k);

    node* new_head = rotate_left(nodeArray[0], n, k);
    printf("Rotated Linked List: \n");
    display(new_head);

    // Free dynamically allocated memory
    node* temp;
    while (new_head != NULL) {
        temp = new_head;
        new_head = new_head->next;
        free(temp);
    }
    free(nodeArray);

    return 0;
}
```

```
Enter the number of nodes: 5
Enter data elements: 1
2
3
4
5
Enter k (number of positions to rotate left): 2
Rotated Linked List:
3->4->5->1->2->NULL

--------------------------------
Process exited after 7.189 seconds with return value 0
Press any key to continue . . .
```

Q2: Given an unsorted linked list of n nodes, remove duplicates from the list.

```cpp
#include<bits/stdc++.h>
using namespace std;
class node{
    public:
    int data;
    node* next;
    public:
    node(int data1,node* next1){
        data=data1;
        next=next1;
    }
    node(int data1){
        data=data1;
        next=nullptr;
    }
};
node* convertarr2LL(vector<int>&arr){
    node* head=new node(arr[0]);
    node* mover=head;
    for(int i=1;i<arr.size();i++){
        node* temp=new node(arr[i]);
        mover->next=temp;
        mover=temp;
    }
return head;
}
```

```
27    void print(node*head){
28        while(head!=NULL){
29            cout<<head->data<<"->";
30            head=head->next;
31        }
32    }
33    node* removeduplicate(node*head){
34        if(head==NULL) return NULL;
35        unordered_set<int> st;
36        node* curr=head;
37        node* prev=nullptr;
38        while(curr!=NULL){
39            if(st.find(curr->data)!=st.end()){
40            prev->next=curr->next;
41            delete curr;
42           }
43           else{
44            st.insert(curr->data);
45            prev=curr;
46           }
47         curr=prev->next;
48        }
49        return head;
50    }
```

```cpp
int main(){
    vector<int> arr={1,2,4,2,1,33,4,4,3};
    node* head=convertarr2LL(arr);
    node* n =removeduplicate(head);
    print(n);
    return 0;
}

```

```
[Running] cd "f:\DSA\" && g++ Assign4_2.cpp -o Assign4_2 && "f:\DSA\"Assign4_2
1->2->4->33->3->
[Done] exited with code=0 in 2.112 seconds
```

Q3: Given a singly linked list of n nodes, detect if it contains a loop or not.

```c
#include<stdio.h>
#include<stdlib.h>

typedef struct Node{
    int data;
    struct Node* next;
}node;

int hascycle(node* head)
{
    node* slow=head;
    node* fast=head;

    while(fast!=NULL&&fast->next!=NULL)
    {
        slow=slow->next;
        fast=fast->next->next;

        if(slow==fast)
        {
            return 1;
        }
    }
    return 0;
```

```c
node* create(int n,int cyclepos)
{
    node* head=NULL;
    node* temp=NULL;
    node* cyclestart=NULL;
    //creating circular linked list
    int i;
    for(i=0;i<n;i++)
    {
        int val;
        node* newnode=(node*)malloc(sizeof(node));
        printf("Enter value for node %d: ", i);
        scanf("%d", &val);


        newnode->data=val;
        newnode->next=NULL;

        if(head==NULL)
        {
            head=newnode;
            temp=head;
        }
```

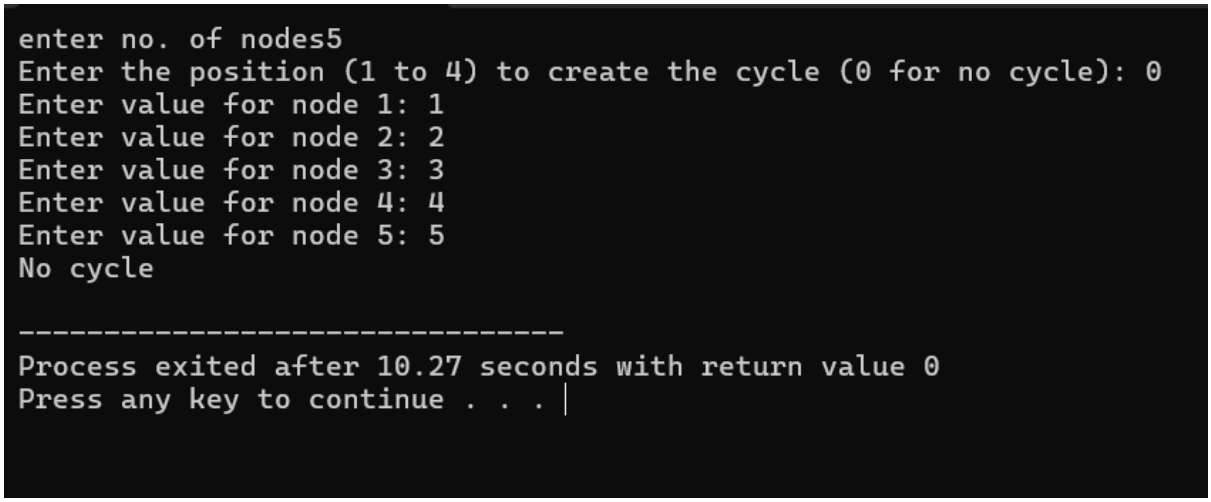
```
50              else
51              {
52                  temp->next=newnode;
53                  temp=temp->next;
54              }
55              if(i==cyclepos)
56              {
57                  cyclestart=newnode;
58              }
59          }
60          if(cyclestart!=NULL)
61          {
62              temp->next=cyclestart;
63          }
64          return head;
65      }
```

```
int main()
{
    int n,pos;
    printf("enter no. of nodes");
    scanf("%d", &n);

    printf("Enter the position (1 to %d) to create the cycle (0 for no cycle): ", n-1);
    scanf("%d", &pos);
    node* head=create(n,pos);
     if (hascycle(head)) {
        printf("Cycle detected\n");
    } else {
        printf("No cycle\n");
    }

    return 0;

}
```



Q4: Write a C/C++ program to implement doubly linked list with the following function

(i) insertAtFirst(&head, new_data): This function should insert the new data/element at the beginning of the linked list.

(ii) insertAtEnd(&head, new_data): This function should insert the new data/element at the end of the linked list

(iii) insertAtMiddle(&head, new_data): This function should insert the new data/element at the middle of the linked list

(iv) insertAfterNode(&head, given_node, new_data): This function should insert the new data/element after the given node in the linked list.

```cpp
#include <bits/stdc++.h>
using namespace std;

class node {
public:
    int data;
    node* next;
    node* back;

    node(int data1, node* next1 = nullptr, node* back1 = nullptr) {
        data = data1;
        next = next1;
        back = back1;
    }
};

// Convert array to DLL
node* convertarr2DLL(vector<int>& arr) {
    if (arr.empty()) return nullptr;

    node* head = new node(arr[0]);
    node* prev = head;
    for (size_t i = 1; i < arr.size(); i++) {
        node* temp = new node(arr[i], nullptr, prev);
        prev->next = temp;
        prev = temp;
    }
```

```
31  // Print DLL
32  void print(node* head) {
33      while (head != NULL) {
34          cout << head->data;
35          if (head->next != NULL) cout << " <-> ";
36          head = head->next;
37      }
38      cout << " -> NULL" << endl;
39  }
40
41  // Insert at beginning
42  node* insertatbegin(node* head, int val) {
43      node* newhead = new node(val, head, nullptr);
44      if (head != nullptr)
45          head->back = newhead;
46      return newhead;
47  }
48
49  // Insert at end
50  node* insertatend(node* head, int val) {
51      if (head == nullptr) return new node(val);
52
53      node* tail = head;
54      while (tail->next != NULL)
55          tail = tail->next;
56
57      node* newnode = new node(val, nullptr, tail);
```

```
58      tail->next = newnode;
59      return head;
60  }
61
62  // Insert at middle
63  node* insertatmiddle(node* head, int val) {
64      if (head == nullptr) return new node(val);
65
66      int cnt = 0;
67      node* temp = head;
68      while (temp != NULL) {
69          cnt++;
70          temp = temp->next;
71      }
72
73      int k = cnt / 2;
74      if (k == 1)
75          return insertatbegin(head, val);
76
77      cnt = 0;
78      temp = head;
79      while (cnt < k) {
80          cnt++;
81          temp = temp->next;
82      }
```

```
84          node* prev = temp->back;
85          node* newnode = new node(val, temp, prev);
86
87          if (prev != nullptr)
88              prev->next = newnode;
89          temp->back = newnode;
90
91          return head;
92      }
93
94      // Insert after a given node
95      void insertafternode(node* temp, int val) {
96          if (temp == nullptr) return;
97
98          node* front = temp->next;
99          node* newnode = new node(val, front, temp);
100
101         temp->next = newnode;
102         if (front != nullptr)
103             front->back = newnode;
104     }
105
106     // Main function
107     int main() {
108         vector<int> arr = {1, 2, 4, 3};
109         node* head = convertarr2DLL(arr);
```

```
  head = insertatbegin(head, 10);
  print(head);
  cout<<"\n";
  head = insertatend(head, 20);
  print(head);
  cout<<"\n";
  head = insertatmiddle(head, 50);
  print(head);
  cout<<"\n";
  insertafternode(head->next, 100);
  print(head);

  return 0;
```

```
[Running] cd "f:\DSA\" && g++ Assign4_4.cpp -o Assign4_4 && "f:\DSA\"Assign4_4
10 <-> 1 <-> 2 <-> 4 <-> 3 -> NULL

10 <-> 1 <-> 2 <-> 4 <-> 3 <-> 20 -> NULL

10 <-> 1 <-> 2 <-> 50 <-> 4 <-> 3 <-> 20 -> NULL

10 <-> 1 <-> 100 <-> 2 <-> 50 <-> 4 <-> 3 <-> 20 -> NULL

[Done] exited with code=0 in 1.498 seconds
```