Shaheed Zulfikar Ali Bhutto Institute of Science & Technology

COMPUTER SCIENCE DEPARTMENT

Total Marks: 7.5

Obtained Marks: _____

DATA STRUCTURE AND ALGORITHM

Lab Report # 04

Submitted To: Mam Tehreen

Submitted By: Hammad Qureshi

Reg. Numbers: 2112114

Question no 1:

a) **Swap nodes in a linked list without swapping data**
b) **Write a function that counts the number of times a given int occurs in a Linked List.**
c) **Write a function Linked list traversal using recursion in c++.**

Code:

Part (a)

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

/* A linked list node */
class Node {
public:
   int data;
   Node* next;
};

/* Function to swap nodes x and y in linked list by
changing links */
void swapNodes(Node** head_ref, int x, int y)
{
   // Nothing to do if x and y are same
   if (x == y)
      return;
```

```
// Search for x (keep track of prevX and CurrX
Node *prevX = NULL ,*currX = *head_ref;
while (currX && currX->data != x)
    {
   prevX = currX;
   currX = currX->next;
}

// Search for y (keep track of prevY and CurrY
Node *prevY = NULL, *currY = *head_ref;
while (currY && currY->data != y) {
   prevY = currY;
   currY = currY->next;
}

// If either x or y is not present, nothing to do
if (currX == NULL || currY == NULL)
   return;

// If x is not head of linked list
if (prevX != NULL)
   prevX->next = currY;
else // Else make y as new head
   *head_ref = currY;

// If y is not head of linked list
if (prevY != NULL)
   prevY->next = currX;
else // Else make x as new head
```

```
        *head_ref = currX;

    // Swap next pointers
    Node* temp = currY->next;
    currY->next = currX->next;
    currX->next = temp;
}

/* Function to add a node at the beginning of List */
void push(Node** head_ref, int new_data)
{
    /* allocate node */
    Node* new_node = new Node();

    /* put in the data */
    new_node->data = 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;
}

/* Function to print nodes in a given linked list */
void printList(Node* node)
{
    while (node != NULL) {
        cout << node->data << " ";
```

```
        node = node->next;
    }
}

/* Driver program to test above function */
int main()
{
    Node* start = NULL;

    /* The constructed linked list is:
    1->2->3->4->5->6->7 */
    push(&start, 7);
    push(&start, 6);
    push(&start, 5);
    push(&start, 4);
    push(&start, 3);
    push(&start, 2);
    push(&start, 1);

    cout << "Linked list before calling swapNodes() ";
    printList(start);

    swapNodes(&start, 4, 3);

    cout << "\nLinked list after calling swapNodes() ";
    printList(start);

    return 0;
}
```

Part (b)

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

/* Link list node */
class Node {
public:
   int data;
   Node* next;
};

/* Given a reference (pointer to pointer) to the head
of a list and an int, push a new node on the front
of the list. */
void push(Node** head_ref, int new_data)
{
   /* allocate node */
   Node* new_node = new Node();

   /* put in the data */
   new_node->data = new_data;

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

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

```
/* Counts the no. of occurrences of a node
(search_for) in a linked list (head)*/
int count(Node* head, int search_for)
{
   Node* current = head;
   int count = 0;
   while (current != NULL) {
      if (current->data == search_for)
         count++;
      current = current->next;
   }
   return count;
}

/* Driver program to test count function*/
int main()
{
   /* Start with the empty list */
   Node* head = NULL;

   /* Use push() to construct below list
   1->2->1->3->1 */
   push(&head, 1);
   push(&head, 3);
   push(&head, 1);
   push(&head, 2);
   push(&head, 1);

   /* Check the count function */
```

```
    cout << "count of 1 is " << count(head, 1);
    return 0;
}
```

Part (c)

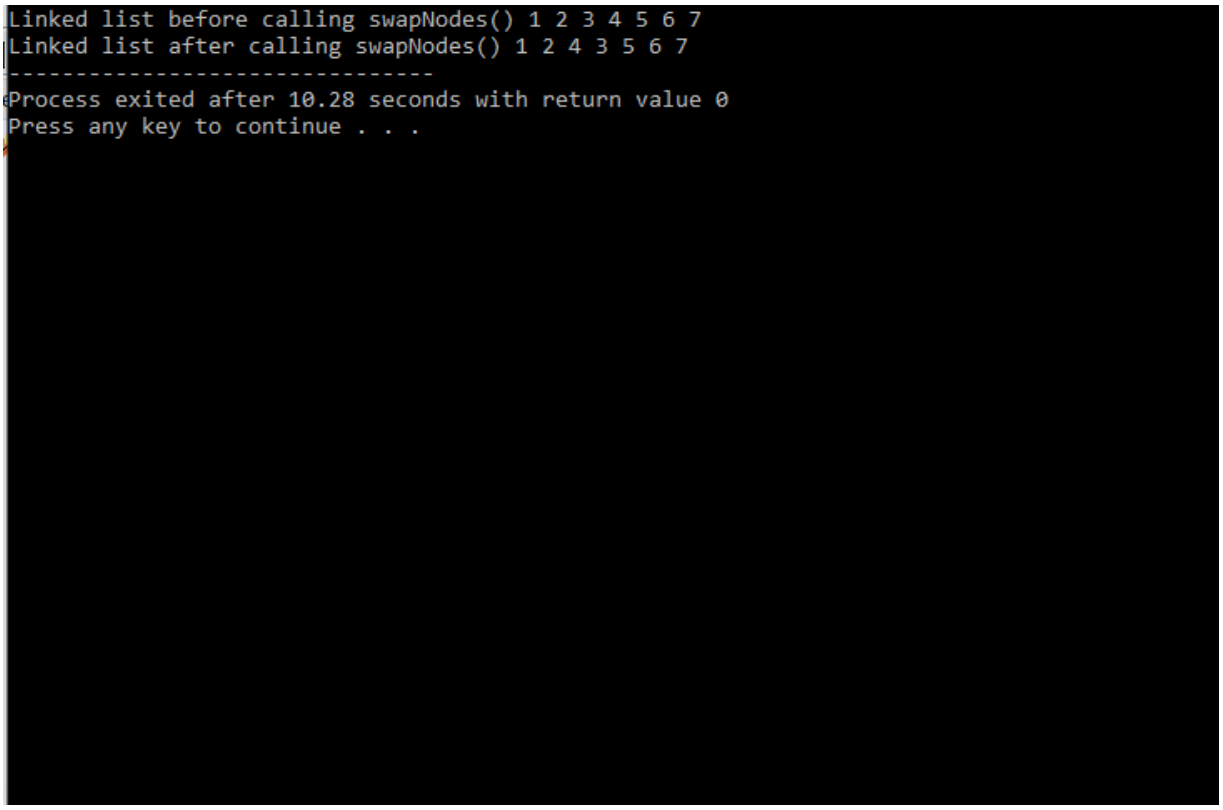
```
#include<iostream>
using namespace std;
struct Node {
    int data;
    Node* next;
};

// Allocates a new node with given data
Node *newNode(int data)
{
    Node *new_node = new Node;
    new_node->data = data;
    new_node->next = NULL;
    return new_node;
}

// Function to insert a new node at the
// end of linked list using recursion.
Node* insertEnd(Node* head, int data)
{
    // If linked list is empty, create a
    // new node (Assuming newNode() allocates
    // a new node with given data)
    if (head == NULL)
        return newNode(data);
```

```
    // If we have not reached end, keep traversing
    // recursively.
    else
        head->next = insertEnd(head->next, data);
    return head;
}

void traverse(Node* head)
{
    if (head == NULL)
        return;

    // If head is not NULL, print current node
    // and recur for remaining list
    cout << head->data << " ";

    traverse(head->next);
}

// Driver code
int main()
{
    Node* head = NULL;
    head = insertEnd(head, 6);
    head = insertEnd(head, 8);
    head = insertEnd(head, 10);
    head = insertEnd(head, 12);
    head = insertEnd(head, 14);
```

```
    traverse(head);
}
```

CONSOLE SCREEN:

Part(a)



Part(b)

```
count of 1 is 3
--------------------------------
Process exited after 7.81 seconds with return value 0
Press any key to continue . . .
```

Part(c)

```
6 8 10 12 14
--------------------------------
Process exited after 7.771 seconds with return value 0
Press any key to continue . . .
```