**Bahria University**

*Lahore Campus*

*Department of Computer Sciences*



DATA STRUCTURES AND ALGORITHMS

**ASSIGNMENT # 01**

**Instructor Name:** Mr. Numan Aslam

**Program:** BSIT 3

**Ali Hassan 03-135211-005**

**Question 1:** Write a GetNth() function that takes a linked list and an integer index and returns the data value stored in the node at that index position.

**Code**

**#include <iostream>**

**using namespace std;**

**class Node**

**{**

**public:**

**int data;**

**Node\* next;**

**};**

**void push(Node\*\* head\_ref, int new\_data)**

**{**

**Node\* new\_node = new Node();**

**new\_node->data = new\_data;**

**new\_node->next = (\*head\_ref);**

**(\*head\_ref) = new\_node;**

**}**

**int GetNth(Node\* head, int index)**

**{**

**Node\* current = head;**

**int count = 0;**

**while (current != NULL) {**

**if (count == index)**

**return (current->data);**

**count++;**

**current = current->next;**

**}**

**}**

**int main()**

**{**

**Node\* head = NULL;**

**push(&head, 1);**

**push(&head, 4);**

**push(&head, 3);**

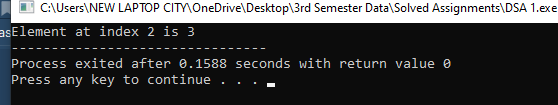
**push(&head, 12);**

**push(&head, 1);**

**cout << "Element at index 2 is " << GetNth(head, 2);**

**return 0;**

**}**



**Question 2:** Write a SortedInsert() function which given a list that is sorted in increasing order, and a single node, inserts the node into the correct sorted position in the list.

**Code**

#include <iostream>

using namespace std;

struct Node

{

int val;

struct Node \*next;

Node(double x)

{

val = x;

next = NULL;

}

};

class Linkedlist

{

public:

Node \*head;

Node \*sorted;

// add node

void push(double val)

{

Node \*newnode = new Node(val);

newnode->next = head;

head = newnode;

}

// insert a new node in a list

void Insert(Node \*newnode)

{

if (sorted == NULL || sorted->val >= newnode->val)

{

newnode->next = sorted;

sorted = newnode;

}

else

{

Node \*current = sorted;

while (current->next != NULL && current->next->val < newnode->val)

{

current = current->next;

}

newnode->next = current->next;

current->next = newnode;

}

}

void SortedInsert(Node \*headref)

{

sorted = NULL;

Node \*current = headref;

while (current != NULL)

{

Node \*next = current->next;

Insert(current);

current = next;

}

head = sorted;

}

void printlist(Node \*head)

{

while (head != NULL)

{

cout << head->val << " ";

head = head->next;

}

}

};

int main()

{

Linkedlist a;

a.head = NULL;

int values[5] = {22, 24, 102, 15, 50};

for (int i = 0; i < 5; i++)

{

a.push(values[i]);

}

cout << "Linked List " << endl;

a.printlist(a.head);

cout << endl;

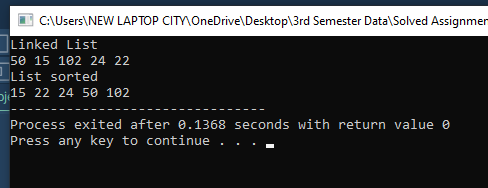
a.SortedInsert(a.head);

cout << "List sorted" << endl;

a.printlist(a.head);

return 0;

}



**Question 3:** Given a list, write a C++ program to print mirror of that list.

Code

#include <iostream>

using namespace std;

struct node

{

int data;

node \*next;

};

void push(node \*\*head\_ref, int data)

{

node \*node;

node = (struct node \*)malloc(sizeof(struct node));

node->data = data;

node->next = (\*head\_ref);

(\*head\_ref) = node;

}

void mirror(struct node \*\*head\_ref)

{

struct node \*temp = NULL;

struct node \*prev = NULL;

struct node \*current = (\*head\_ref);

while (current != NULL)

{

temp = current->next;

current->next = prev;

prev = current;

current = temp;

}

(\*head\_ref) = prev;

}

void printnodes(node \*head)

{

while (head != NULL)

{

cout << head->data << " ";

head = head->next;

}

}

int main()

{

node \*head = NULL;

push(&head, 300);

push(&head, 10);

push(&head, 200);

push(&head, 30);

cout << "Linked List:" << endl;

printnodes(head);

mirror(&head);

cout << endl;

cout << "Mirrored List:" << endl;

printnodes(head);

return 0;

}

