

| **Title: Implementation of Linked List** |
| --- |

**Objective:** To understand the use of linked list as data structures for various application.

**Expected Outcome of Experiment:**

| **CO** | **Outcome** |
| --- | --- |
| **CO 2** | Apply linear and non-linear data structure in application development. |

**Books/ Journals/ Websites referred:**

**Mam’s ppt**

**Introduction:**

Define Linked List

A linked list is a sequence of data structures, which are connected together via links. Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list is the second most-used data structure after array.

**Types of linked list:**

* Singly linked lists
* Doubly linked lists
* Circular linked lists
* Circular doubly linked lists

**Algorithm for creation, insertion, deletion, traversal and searching an element in assigned linked list type:**

**Implementation of an application using linked list:**

**Code:**

**#include <stdlib.h>**

**#include <iostream>**

**using namespace std;**

**struct Node {**

**int data;**

**struct Node\* next;**

**};**

**void insertAtBeginning(struct Node\*\* head\_ref, int new\_data) {**

**struct Node\* new\_node = (struct Node\*)malloc(sizeof(struct Node));**

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

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

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

**}**

**void insertAfter(struct Node\* prev\_node, int new\_data) {**

**if (prev\_node == NULL) {**

**cout << "the given previous node cannot be NULL";**

**return;**

**}**

**struct Node\* new\_node = (struct Node\*)malloc(sizeof(struct Node));**

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

**new\_node->next = prev\_node->next;**

**prev\_node->next = new\_node;**

**}**

**void insertAtEnd(struct Node\*\* head\_ref, int new\_data) {**

**struct Node\* new\_node = (struct Node\*)malloc(sizeof(struct Node));**

**struct Node\* last = \*head\_ref; /\* used in step 5\*/**

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

**new\_node->next = NULL;**

**if (\*head\_ref == NULL) {**

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

**return;**

**}**

**while (last->next != NULL) last = last->next;**

**last->next = new\_node;**

**return;**

**}**

**void deleteNode(struct Node\*\* head\_ref, int key) {**

**struct Node \*temp = \*head\_ref, \*prev;**

**if (temp != NULL && temp->data == key) {**

**\*head\_ref = temp->next;**

**free(temp);**

**return;**

**}**

**while (temp != NULL && temp->data != key) {**

**prev = temp;**

**temp = temp->next;**

**}**

**if (temp == NULL) return;**

**prev->next = temp->next;**

**free(temp);**

**}**

**bool searchNode(struct Node\*\* head\_ref, int key) {**

**struct Node\* current = \*head\_ref;**

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

**if (current->data == key) return true;**

**current = current->next;**

**}**

**return false;**

**}**

**void sortLinkedList(struct Node\*\* head\_ref) {**

**struct Node \*current = \*head\_ref, \*index = NULL;**

**int temp;**

**if (head\_ref == NULL) {**

**return;**

**} else {**

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

**index = current->next;**

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

**if (current->data > index->data) {**

**temp = current->data;**

**current->data = index->data;**

**index->data = temp;**

**}**

**index = index->next;**

**}**

**current = current->next;**

**}**

**}**

**}**

**void printList(struct Node\* node) {**

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

**cout << node->data << " ";**

**node = node->next;**

**}**

**}**

**int main() {**

**struct Node\* head = NULL;**

**int n,a,b,c,d,e;**

**while(1){**

**cout<<"Select an option: "<<endl;**

**cout<<"\n1. insert at beginning \n";**

**cout<<"2.print \n";**

**cout<<"3. insert at end. \n";**

**cout<<"4. Insert after a point, \n";**

**cout<<"5. delete an element. \n";**

**cout<<"6.Search \n";**

**cin>>n;**

**switch(n)**

**{**

**case 1 : cout<<"Enter element: \n";**

**cin>>a;**

**insertAtBeginning(&head, a);**

**break;**

**case 2 : cout << "Linked list: \n";**

**sortLinkedList(&head);**

**printList(head);**

**break;**

**case 3 : cout<<"Enter element: \n";**

**cin>>b;**

**insertAtEnd(&head, b);**

**break;**

**case 4 : cout<<"enter Element : \n";**

**cin>>c;**

**insertAfter(head->next, c);**

**break;**

**case 5 : cout<<"Enter element u wanna delete: \n";**

**cin>>d;**

**deleteNode(&head, d);**

**break;**

**case 6 : cout<<"Enter element u wanna search: \n";**

**cin>>e;**

**if (searchNode(&head, e)) {**

**cout << " Found";**

**} else {**

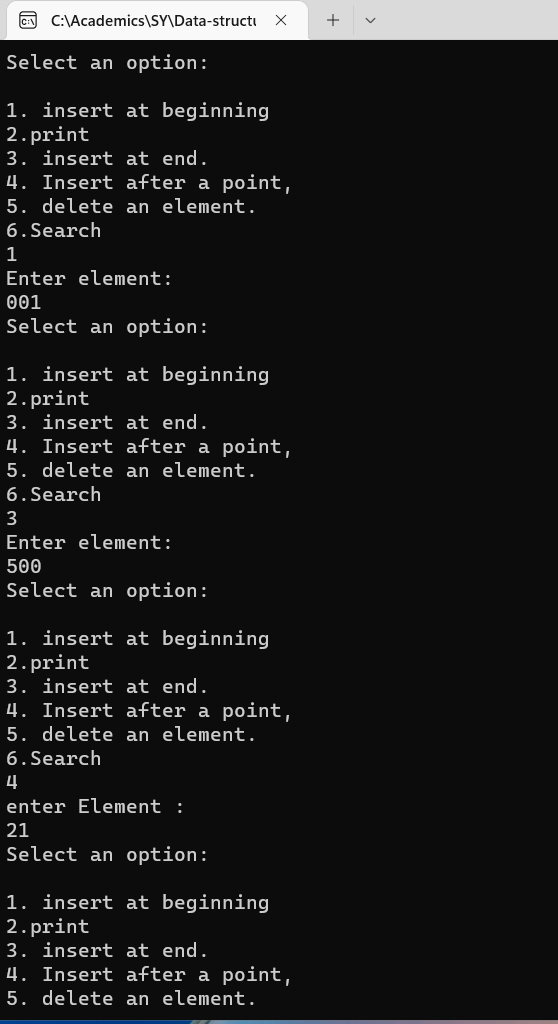
**cout << "Not Found";**

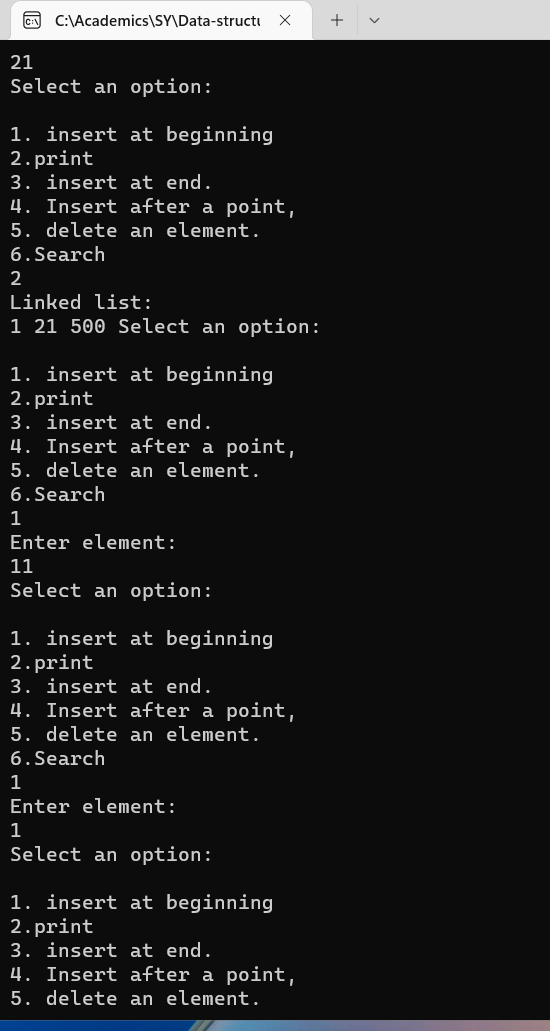
**}**

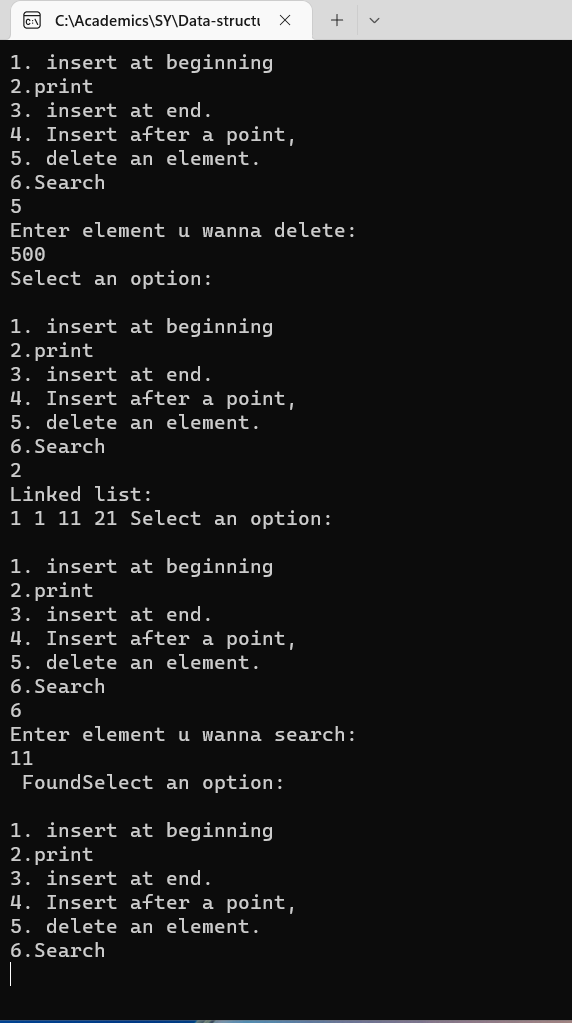
**break;**

**}}}**

**Output:**

****

****

****

**Conclusion:-**

**Linked list was implemented successfully and various operations in linked list were also performed.**

**Post lab questions:**

1. Compare and contrast SLL and DLL

