Batch:A3 Roll No.:16010121051

Experiment No. 2

Grade: AA / AB / BB / BC / CC / CD /DD

Title: Implementation of different operations on Linked List – creation, insertion, deletion, traversal, searching an element

Objective: To understand the advantage of linked list over other structures like arrays in implementing the general linear list

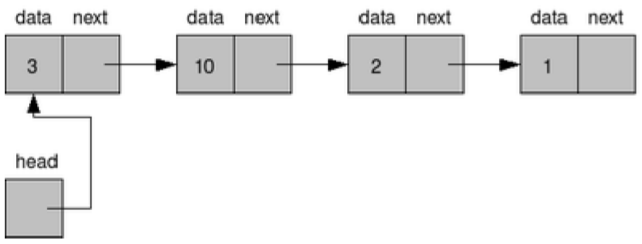
Expected Outcome of Experiment:

CO	Outcome
CO 1	To understand the advantage of linked list over other structures like arrays in implementing the general linear list

Books/ Journals/ Websites referred:

Introduction:

A linear list is a list where each element has a unique successor. There are four common operations associated with linear list: insertion, deletion, retrieval, and traversal. Linear list can be divided into two categories: general list and restricted list. In general list the data can be inserted or deleted without any restriction whereas in restricted list there is restrictions for these operations. Linked list and arrays are commonly used to implement general linear list. A linked list is simply a chain of structures which contain a pointer to the next element. It is dynamic in nature. Items may be added to it or deleted from it at will.



A list item has a pointer to the next element, or to NULL if the current element is the tail (end of the list). This pointer points to a structure of the same type as itself. This Structure that contains elements and pointers to the next structure is called a Node.

Related Theory: -

In computer science, a linked list is a linear collection of data elements, whose order is not given by their physical placement in memory. Instead, each element points to the next. It is a data structure consisting of a collection of nodes which together represent a sequence. In its most basic form, each node contains: data, and a reference to the next node in the sequence. This structure allows for efficient insertion or removal of elements from any position in the sequence during iteration.

Like arrays, Linked List is a linear data structure. Unlike arrays, linked list elements are not stored at contiguous location; the elements are linked using pointers

Linked List

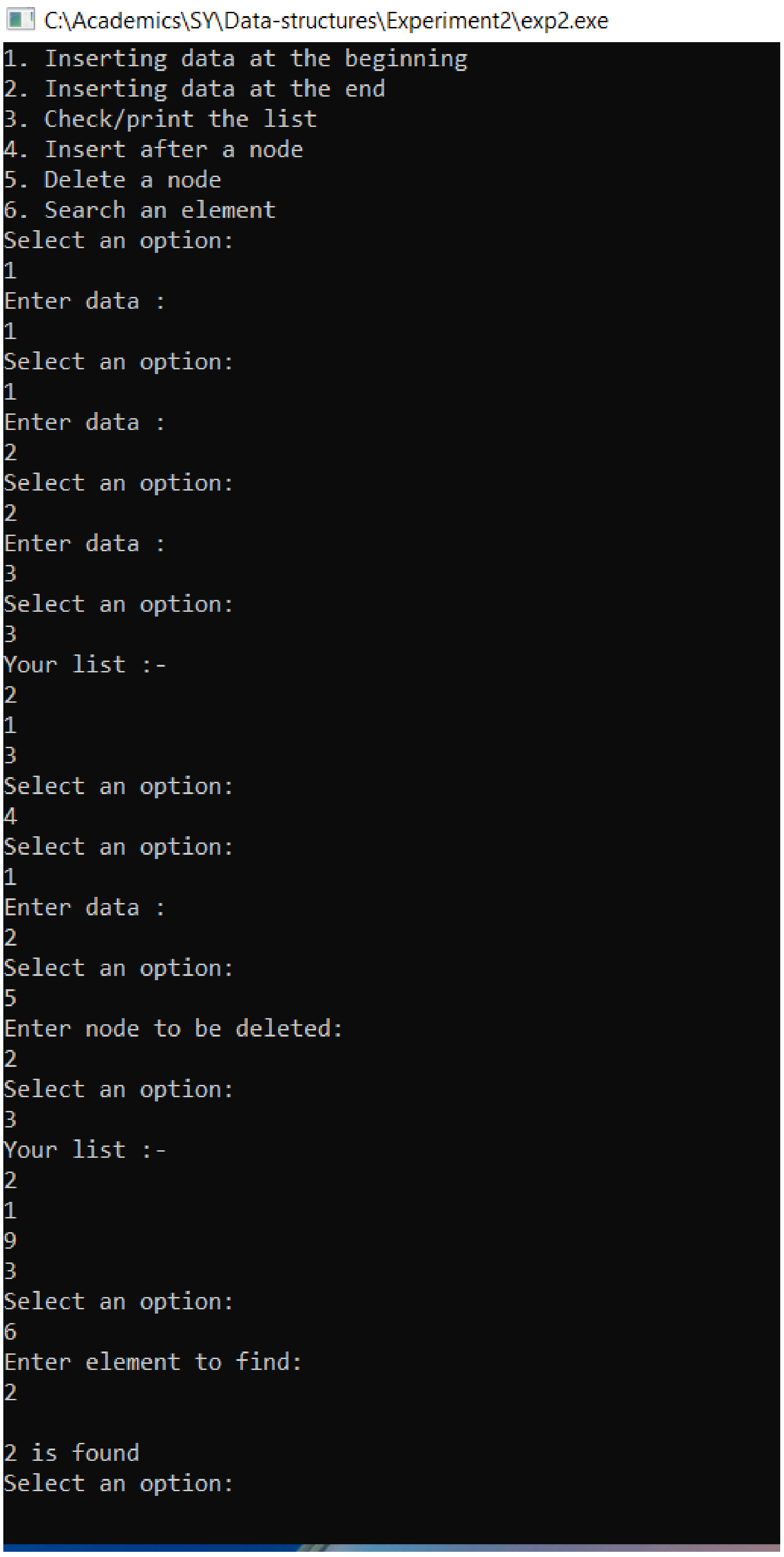
Algorithm for creation, insertion, deletion, traversal and searching an element in assigned linked list type:

Implementation:

```
//linked list and its operations
#include<iostream>
using namespace std;

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

void insertAtBeginning(struct Node** test,int new_data) {

  struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));

  new_node->data = new_data;
  new_node->next = (*test);

  (*test) = new_node;
}
void insertAtEnd(struct Node** test, int new_data) {
  struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
  struct Node* last = *test;

  new_node->data = new_data;
  new_node->next = NULL;

  if (*test == NULL) {
  *test = new_node;
  return;
  }
  while (last->next != NULL) last = last->next;

  last->next = new_node;
  return;
}

void insertAfter(struct Node* prev_node, int new_data) {
  if (prev_node == NULL) {
  cout << "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 deleteNode(struct Node** test, int key) {
  struct Node *temp = *test, *prev;

  if (temp != NULL && temp->data == key) {
  *test = 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** test, int key) {
  struct Node* current = *test;

  while (current != NULL) {
  if (current->data == key) return true;
  current = current->next;
  }
  return false;
}
void printList(struct Node* node) {

  while (node != NULL) {
    cout << node->data<<endl ;
    node = node->next;
  }}

int main()
{
int n,a,b,c,d;
struct Node* head = NULL;
cout<<"1. Inserting data at the beginning"<<endl;
cout<<"2. Inserting data at the end"<<endl;
cout<<"3. Check/print the list"<<endl;
cout<<"4. Insert after a node "<<endl;
cout<<"5. Delete a node"<<endl;
cout<<"6. Search an element"<<endl;
```

```
while(1){
cout<<"Select an option: "<<endl;
cin>>n;

switch(n)
        {
                case 1: cout<<"Enter data :"<<endl;
                                cin>>a;
                                insertAtBeginning(&head,a);
                                break;
                case 3:cout<<"Your list :- "<<endl;
                          printList(head);
                          break;
                case 2: cout<<"Enter data :"<<endl;
                                cin>>b;
                                insertAtEnd(&head,b);
                                break;
        case 4:  insertAfter(head->next, 9);
                break;
        case 5: cout<<"Enter node to be deleted: "<<endl;
                cin>>c;
                deleteNode(&head, c);
                break;
        case 6:cout<<"Enter element to find: "<<endl;
            cin>>d;
                                 if (searchNode(&head, d)) {
                                 cout << endl << d << " is found"<<endl;
                                 } else {
                                 cout << endl <<d<< " is not found"<<endl;
                                 }
                        }
}
return 0;}
```

Output:



```
C:\Academics\SY\Data-structures\Experiment2\exp2.exe
1. Inserting data at the beginning
2. Inserting data at the end
3. Check/print the list
4. Insert after a node
5. Delete a node
6. Search an element
Select an option:
1
Enter data :
1
Select an option:
1
Enter data :
2
Select an option:
2
Enter data :
3
Select an option:
3
Your list :-
2
1
3
Select an option:
4
Select an option:
1
Enter data :
2
Select an option:
5
Enter node to be deleted:
2
Select an option:
3
Your list :-
2
1
9
3
Select an option:
6
Enter element to find:
2

2 is found
Select an option:
```

Conclusion:-

Successfully completed stack operations using both array and linked list.

Post lab questions:

1. Write the differences between linked list and linear array

ARRAY	LINKED LIST
An array is a grouping of data elements of equivalent data type.	A linked list is a group of entities called a node. The node includes two segments: data and address.
It stores the data elements in a contiguous memory zone.	It stores elements randomly, or we can say anywhere in the memory zone.
In the case of an array, memory size is fixed, and it is not possible to change it during the run time.	In the linked list, the placement of elements is allocated during the run time.
The elements are not dependent on each other	The data elements are dependent on each other.
The memory is assigned at compile time.	The memory is assigned at run time.
It is easier and faster to access the element in an array.	In a linked list, the process of accessing elements takes more time.
In the case of an array, memory utilization is ineffective.	In the case of the linked list, memory utilization is effective.
When it comes to executing any operation like insertion, deletion, array takes more time.	When it comes to executing any operation like insertion, deletion, the linked list takes less time.

2. Name some applications which uses linked list.

- Polynomial Manipulation
- Addition of long +ve integers
- Symbol table creation
- Memory management