Name: KAUSHAL VASHISTH Roll Number: 18ETCS002147

Laboratory 8

1. Questions

    1. Implement a linked list and perform following operations.
        i. Insert a node before and after a given node
        ii. Delete a node before and after a given node
    2. Implement a linked list to create and print a binary tree.

2. Program

1.1)Insert a node before and after a given node

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

/* A structure of linked list node */
struct node {
  int data;
  struct node *next;
} *head;

void initialize(){
    head = NULL;
}

void insert(int num) {
    /* Create a new Linked List node */
    struct node* newNode = (struct node*) malloc(sizeof(struct node));
    newNode->data  = num;
    /* Next pointer of new node will point to head node of linked list  */
    newNode->next = head;
    /* make new node as new head of linked list */
    head = newNode;
    printf("Inserted Element : %d\n", num);
}
```

```c

void insertAfter(struct node* prevNode, int num) {
    /* Input validation */
    if (prevNode == NULL) {
        printf("Error : Invalid node pointer !!!\n");
        return;
    }

    /* creates a new node */
    struct node* newNode =(struct node*) malloc(sizeof(struct node));
    newNode->data  = num;
    /* Set Next pointer of newNode to next pointer of nodePtr */
    newNode->next = prevNode->next;
    /* Set next pointer of prevNode to newNode */
    prevNode->next = newNode;
}

/*
Prints a linked list from head node till tail node
*/
void printLinkedList(struct node *nodePtr) {
     printf("\nLinked List\n");
   while (nodePtr != NULL) {
      printf("%d", nodePtr->data);
      nodePtr = nodePtr->next;
      if(nodePtr != NULL)
          printf("-->");
   }
}

```

```c
int main() {
    initialize();
    /* Creating a linked List*/
    insert(99);
    insert(78);
    insert(55);
    insert(45);
    printLinkedList(head);
    /* Inserting a node after third node(4) from head */
    insertAfter(head->next->next, 8);
    printf("\n\nAfter Insertion\n");
    printLinkedList(head);
    return 0;
}

```

1.2)Delete a node before and after a given node

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

// A linked list node
struct Node
{
   int data;
   struct Node *next;
};

void push(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;
}
```

Name: KAUSHAL VASHISTH Roll Number: 18ETCS002147

```
19  void deleteNode(struct Node **head_ref, int position)
20  {
21  // If linked list is empty
22  if (*head_ref == NULL)
23     return;
24
25  struct Node* temp = *head_ref;
26
27     // If head needs to be removed
28     if (position == 0)
29     {
30        *head_ref = temp->next; // Change head
31        free(temp);         // free old head
32        return;
33     }
34  int i=0;
35     for (temp!=NULL && i<position-1; i++;){
36        temp = temp->next;
37     }
38     if (temp == NULL || temp->next == NULL)
39        return;
40     struct Node *next = temp->next->next;
41
42     // Unlink the node from linked list
43     free(temp->next); // Free memory
44
45     temp->next = next; // Unlink the deleted node from list
46  }
```

```
47
48  void printList(struct Node *node)
49  {
50      while (node != NULL)
51      {
52          printf(" %d ", node->data);
53          node = node->next;
54      }
55  }
56  int main()
57  {
58      /* Start with the empty list */
59      struct Node* head = NULL;
60
61      push(&head, 56);
62      push(&head, 145);
63      push(&head, 37);
64      push(&head, 28);
65      push(&head, 80);
66
67      puts("Created Linked List: ");
68      printList(head);
69      deleteNode(&head, 4);
70      puts("\nLinked List after Deletion at position 4: ");
71      printList(head);
72      return 0;
73  }
```

Q2) Implement a linked list to create and print a binary tree.

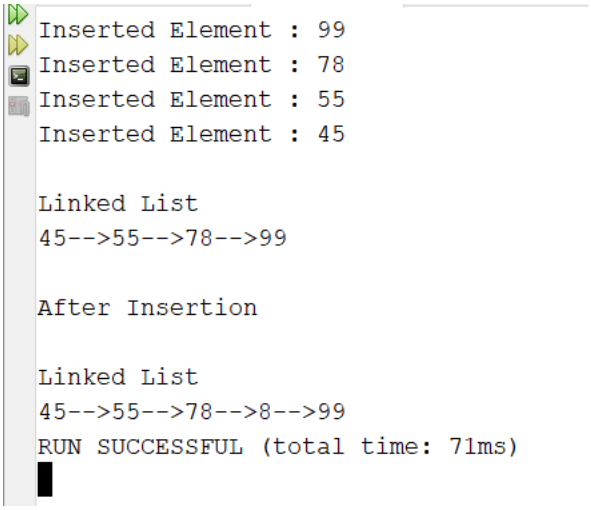
```
1  #include <stdio.h>
2  #include <malloc.h>
3
4  struct node {
5      struct node * left;
6      char data;
7      struct node * right;
8  };
9
10 struct node *constructTree( int );
11 void inorder(struct node *);
12
13 char array[ ] = { 'e', 'f', 'g', 'h', 'i', 'j', 'k', '\0', '\0', 'l' };
14 int leftcount[ ] = {  1,   3,   5,   -1,   9,  -1,  -1,   -1,   -1,  -1 };
15 int rightcount[ ] = {  2,   4,   6,   -1,  -1,  -1,  -1,   -1,   -1,  -1 };
16
```

```
17  void main() {
18      struct node *root;
19      root = constructTree( 0 );
20      printf("In-order Traversal: \n");
21      inorder(root);
22  }
23
24  struct node * constructTree( int index ) {
25      struct node *temp = NULL;
26      if (index != -1) {
27          temp = (struct node *)malloc( sizeof ( struct node ) );
28          temp->left = constructTree( leftcount[index] );
29          temp->data = array[index];
30          temp->right = constructTree( rightcount[index] );
31      }
32      return temp;
33  }
```

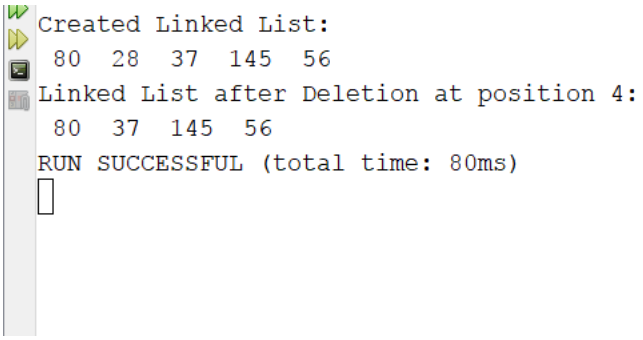
```
34
35  void inorder( struct node *root ) {
36      if (root != NULL) {
37          inorder(root->left);
38          printf("%c\t", root->data);
39          inorder(root->right);
40      }
41  }
42
```

3. Presentation of Results

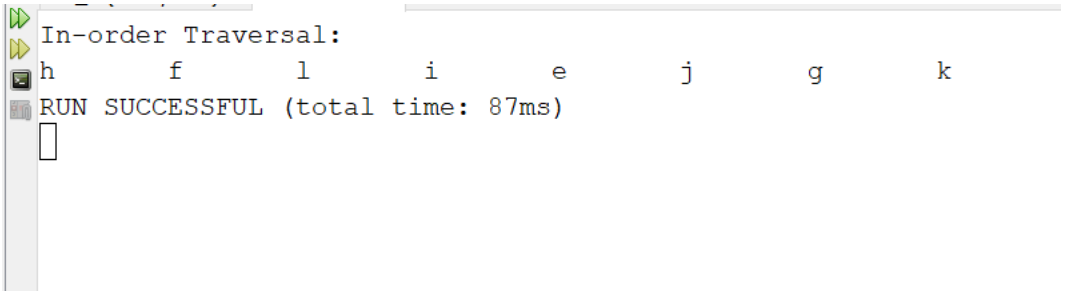
1.1)Insert a node before and after a given node



1.2) delete a node before and after a given node



Q2:-



4. Conclusions

All the programs have been executed successfully.

Linked lists and binary tree have been revised successfully.