BHAVLEEN KAUR
3EIC-2
102155010

LAB ASSIGNMENT 4 (a)
Linked-list

1) Write a program to implement single Link List (with function insertion at first node, insertion at last node, insertion and deletion at middle node and traversing of link list after each node insertion).

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

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

void insertAtBeginning(struct Node** head, int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = *head;
    *head = newNode;
}
void deleteAtBeginning(struct Node** head) {
    if (*head == NULL) {
        printf("Linked list is already empty.\n");
        return;
    }
    struct Node* temp = *head;
    *head = (*head)->next;
    free(temp);
}
void deleteAtEnd(struct Node** head) {
    if (*head == NULL) {
        printf("Linked list is already empty.\n");
        return;
    }
    struct Node* temp = *head;
    struct Node* prev = NULL;
    while (temp->next != NULL) {
        prev = temp;
        temp = temp->next;
    }
    if (prev == NULL) {
        *head = NULL;
    } else {
        prev->next = NULL;
    }
    free(temp);
```

```c
void deleteAtPosition(struct Node** head, int position) {
    if (*head == NULL) {
        printf("Linked list is already empty.\n");
        return;
    }
    struct Node* temp = *head;
    struct Node* prev = NULL;
    if (position == 1) {
        *head = temp->next;
        free(temp);
        return;
    }
    for (int i = 1; temp != NULL && i < position; i++) {
        prev = temp;
        temp = temp->next;
    }
    if (temp == NULL) {
        printf("Invalid position.\n");
        return;
    }
    prev->next = temp->next;
    free(temp);
}
void displayList(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}
int main() {
    struct Node* head = NULL;
    insertAtBeginning(&head, 3);
    insertAtBeginning(&head, 2);
    insertAtBeginning(&head, 1);
    printf("Linked List: ");
    displayList(head);
    deleteAtBeginning(&head);
    printf("Linked List after deletion at the beginning: ");
```

BHAVLEEN KAUR
3EIC-2
102155010

```
82      displayList(head);
83      deleteAtEnd(&head);
84      printf("Linked List after deletion at the end: ");
85      displayList(head);
86      deleteAtPosition(&head, 1);
87      printf("Linked List after deletion at a specific position: ");
88      displayList(head);
89      return 0;
90  }
```

```
/tmp/hkkGhn41JT.o
Linked List: 1 2 3
Linked List after deletion at the beginning: 2 3
Linked List after deletion at the end: 2
```

2) Write a program to implement single Link List (with function deletion at first node, deletion at last node, and deletion at middle node and traversing of link list after each node deletion).

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   struct Node {
4       int data;
5       struct Node* next;
6   };  int, inserts a new node on the front of the
7   list. */
8   void push(struct Node** head_ref, int new_data)
9   {
10      struct Node* new_node
11          = (struct Node*)malloc(sizeof(struct Node));
12      new_node->data = new_data;
13      new_node->next = (*head_ref);
14      (*head_ref) = new_node;
15  }
16  void deleteNode(struct Node** head_ref, int key)
17  {
18      struct Node *temp = *head_ref, *prev;
19      if (temp != NULL && temp->data == key) {
20          *head_ref = temp->next;
21          free(temp);
22          return;
23      }
24      while (temp != NULL && temp->data != key) {
25          prev = temp;
26          temp = temp->next;
27      }
28      if (temp == NULL)
29          return;
30      prev->next = temp->next;
31  }
32  void printList(struct Node* node)
33  {
34      while (node != NULL) {
35          printf(" %d ", node->data);
36          node = node->next;
37      }
38  }
39  int main()
40  {
41      struct Node* head = NULL;
42      push(&head, 7);
43      push(&head, 1);
44      push(&head, 3);
45      push(&head, 2);
46      puts("Created Linked List: ");
47      printList(head);
48      deleteNode(&head, 1);
49      puts("Linked List after Deletion of 1: ");
50      printList(head);
51      return 0;
52  }
53
```