Program 5

Write A Program to Implement Singly Linked List with following operations
a) Create a linked list.
b) Deletion of first element, specified element and last element in the list.
c) Display the contents of the linked list.

Code:

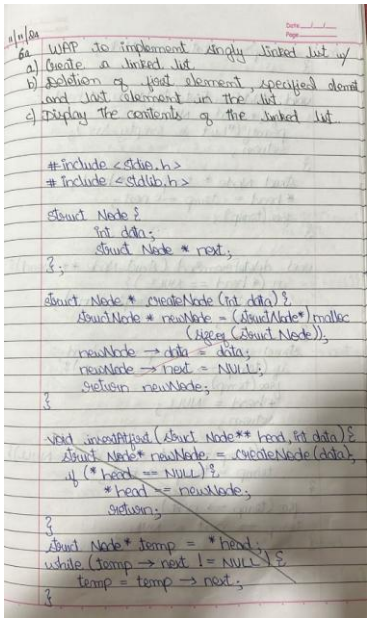
```
#include <stdio.h>
#include <stdlib.h>

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

struct Node* createNode(int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->next = NULL;
   return newNode;
};

void insertatfirst(struct Node** head, int data){
   struct Node* newnode =createNode(data);
   newnode->next = *head;
   *head = newnode;
}

void deleteFirst(struct Node** head) {
   if (*head == NULL) {
      printf("The list is empty.\n");
      return;
   }
   struct Node* temp = *head;
   *head = (*head)->next;
   free(temp);
}

void deleteElement(struct Node** head, int key) {
   if (*head == NULL) {
      printf("The list is empty.\n");
      return;
   }
   struct Node *temp = *head, *prev = NULL;
   if (temp != NULL && temp->data == key) {
      *head = temp->next;
      free(temp);
      return;
```

```
    }
    while (temp != NULL && temp->data != key) {
        prev = temp;
        temp = temp->next;
    }
    if (temp == NULL) {
        printf("Element %d not found.\n", key);
        return;
    }
    prev->next = temp->next;
    free(temp);
}

void deleteLast(struct Node** head) {
    if (*head == NULL) {
        printf("The list is empty.\n");
        return;
    }
    struct Node *temp = *head, *prev = NULL;
    if (temp->next == NULL) {
        *head = NULL;
        free(temp);
        return;
    }
    while (temp->next != NULL) {
        prev = temp;
        temp = temp->next;
    }
    prev->next = NULL;
    free(temp);
}

void displayList(struct Node* head) {
    if (head == NULL) {
        printf("The list is empty.\n");
        return;
    }
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    struct Node* head = NULL;
    int choice, value;

    while (1) {
```

```
        printf("\nMenu:\n");
        printf("1. Insert element at the end\n 2. Delete first element\n 3.Delete specified element\n 4.Delete last element\n 5.Display list\n 6.Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value to insert: ");
                scanf("%d", &value);
                insertatfirst(&head, value);
                break;
            case 2:
                deleteFirst(&head);
                break;
            case 3:
                printf("Enter value to delete: ");
                scanf("%d", &value);
                deleteElement(&head, value);
                break;
            case 4:
                deleteLast(&head);
                break;
            case 5:
                displayList(head);
                break;
            case 6:
                exit(0);
            default:
                printf("Invalid choice.\n");
        }
    }
    return 0;
}
```

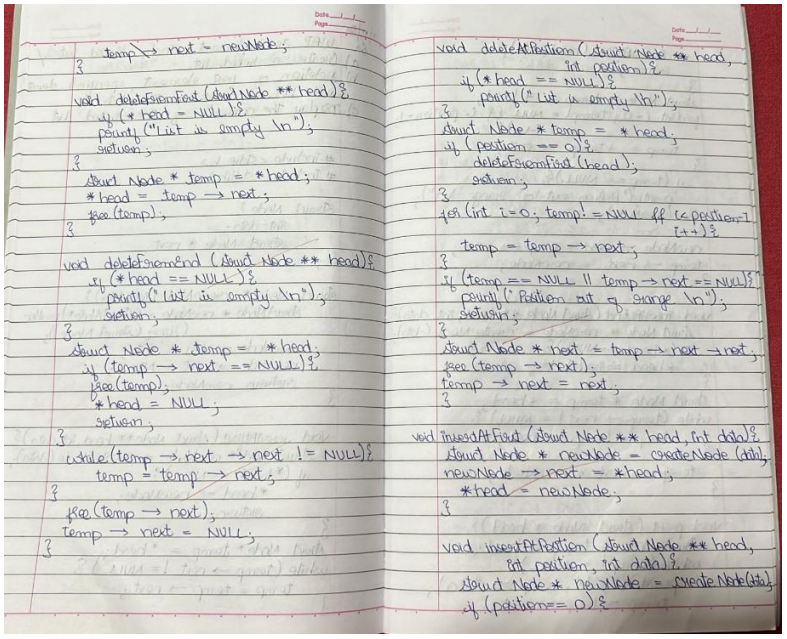
```
Menu:
1. Insert element at the end
 2. Delete first element
 3.Delete specified element
 4.Delete last element
 5.Display list
 6.Exit
Enter your choice: 2

Menu:
1. Insert element at the end
 2. Delete first element
 3.Delete specified element
 4.Delete last element
 5.Display list
 6.Exit
Enter your choice: 5
38 -> 23 -> 14 -> NULL

Menu:
1. Insert element at the end
 2. Delete first element
 3.Delete specified element
 4.Delete last element
 5.Display list
 6.Exit
Enter your choice: 4
```

```
Menu:
1. Insert element at the end
 2. Delete first element
 3.Delete specified element
 4.Delete last element
 5.Display list
 6.Exit
Enter your choice: 5
38 -> 23 -> NULL

Menu:
1. Insert element at the end
 2. Delete first element
 3.Delete specified element
 4.Delete last element
 5.Display list
 6.Exit
Enter your choice: 3
Enter value to delete: 23

Menu:
1. Insert element at the end
 2. Delete first element
 3.Delete specified element
 4.Delete last element
 5.Display list
 6.Exit
Enter your choice: 5
38 -> NULL
```



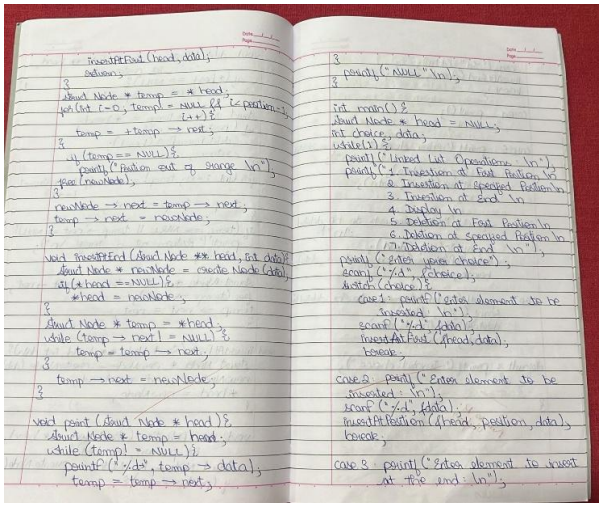
6a WAP to implement singly linked list w/
a) Create a linked list
b) Deletion of your element, specified domit and last element in the list
c) Display the contents of the linked list.

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

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

struct Node * createNode (int data) {
    struct Node * newNode = (struct Node*) malloc
                (sizeof (struct Node));
    newNode -> data = data;
    newNode -> next = NULL;
    return newNode;
}

void insertAtFirst (struct Node** head, int data) {
    struct Node* newNode = createNode (data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    struct Node* temp = *head;
    while (temp -> next != NULL) {
        temp = temp -> next;
    }
```

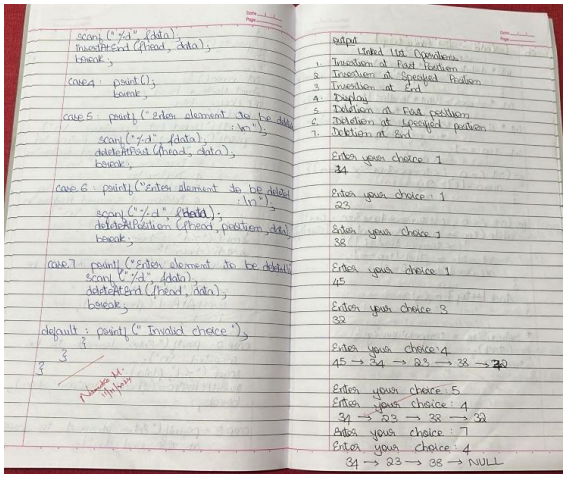


```
    temp -> next = newNode;
}

void deleteFromFirst (struct Node ** head) {
    if (* head = NULL) {
    printf ("List is empty \n");
    return;
}
    struct Node * temp = *head;
    *head = temp -> next;
    free (temp);
}

void deleteFromEnd (struct Node ** head) {
    if (*head == NULL) {
    printf (" List is empty \n");
    return;
}
    struct Node * temp = *head;
    if (temp -> next == NULL) {
    free (temp);
    *head = NULL;
    return;
}
    while (temp -> next -> next != NULL) {
        temp = temp -> next;
}
    free (temp -> next);
    temp -> next = NULL;
}
```

```
void deleteAtPosition (struct Node ** head,
                       int position) {
    if (* head == NULL) {
    printf (" List is empty \n");
}
    struct Node * temp = *head;
    if (position == 0) {
        deleteFromFirst (head);
        return;
}
    for (int i=0; temp! = NULL && i<position-1
                                    i++) {
        temp = temp -> next;
}
    if (temp == NULL || temp -> next == NULL) {
        printf (" Position out of range \n");
        return;
}
    struct Node * next = temp -> next -> next;
    free (temp -> next);
    temp -> next = next;
}

void insertAtFirst (struct Node ** head, int data) {
    struct Node * newNode = createNode (data);
    newNode -> next = *head;
    *head = newNode;
}

void insertAtPosition (struct Node ** head,
                       int position, int data) {
    struct Node * newNode = createNode (data);
    if (position == 0) {
```



```
        insertAtFirst (head, data);
        return;
}
    struct Node * temp = *head;
    for (int i=0; temp! = NULL && i<position-1;
                                    i++) {
        temp = temp -> next;
}
    if (temp == NULL) {
        printf ("Position out of range \n");
        free (newNode);
}
    newNode -> next = temp -> next;
    temp -> next = newNode;
}

void insertAtEnd (struct Node ** head, int data) {
    struct Node * newNode = create Node (data);
    if (*head == NULL) {
        *head = newNode;
        return;
}
    struct Node * temp = *head;
    while (temp -> next != NULL) {
        temp = temp -> next;
}
    temp -> next = newNode;
}

void print (struct Node * head) {
    struct Node * temp = head;
    while (temp! = NULL) {
        printf ("%d", temp -> data);
        temp = temp -> next;
}
    printf ("NULL \n");
}

int main () {
    struct Node * head = NULL;
    int choice, data;
    while (1) {
        printf ("Linked List Operations \n");
        printf (" 1. Insertion at Fast Position \n
                2. Insertion at Specified Position \n
                3. Insertion at End \n
                4. Display \n
                5. Deletion at Fast Position \n
                6. Deletion at Specified Position \n
                7. Deletion at End \n");
        printf ("Enter your choice");
        scanf ("%d", &choice);
        switch (choice) {
            case 1: printf ("Enter element to be
                    inserted: \n");
                scanf ("%d", &data);
                insertAtFirst (&head, data);
                break;

            case 2: printf ("Enter element to be
                    inserted: \n");
                scanf ("%d", &data);
                insertAtPosition (&head, position, data);
                break;

            case 3: printf ("Enter element to insert
                    at the end: \n");
```



```
                scanf ("%d", &data);
                insertAtEnd (&head, data);
                break;

            case 4: print ();
                break;

            case 5: printf ("Enter element to be del
                                    : \n");
                scanf ("%d", &data);
                deleteAtFirst (&head, data);
                break;

            case 6: printf ("Enter element to be delet
                                    : \n");
                scanf ("%d", &data);
                deleteAtPosition (&head, position, dat
                break;

            case 7: printf ("Enter element to be delet
                scanf ("%d", &data);
                deleteAtEnd (&head, data);
                break;

            default : printf (" Invalid choice ");
        }
    }
}
```

Output

```
Linked List Operations
1. Insertion at Fast Position
2. Insertion at Specified Position
3. Insertion at End
4. Display
5. Deletion at Fast position
6. Deletion at specified position
7. Deletion at End

Enter your choice 1
34

Enter your choice 1
23

Enter your choice 1
38

Enter your choice 1
45

Enter your choice 3
32

Enter your choice 4
45 → 34 → 23 → 38 → 32

Enter your choice 5
Enter your choice 4
34 → 23 → 38 → 32
Enter your choice 7
Enter your choice 4
34 → 23 → 38 → NULL
```