**Implementation of Single Linked List**

// Implementing linked list

#include <stdio.h>

#include <stdlib.h>

struct node

{

  int data;

  struct node \*link;

};

struct node \*head = NULL;

int count = 0;

void insertAtBegin(int x);

void insertAtEnd(int x);

void insertAtPosition(int x, int pos);

void deleteAtBegin();

void deleteAtEnd();

void deleteAtPosition(int pos);

void display();

void search(int key);

int main()

{

  insertAtBegin(10);

  insertAtBegin(20);

  insertAtBegin(30);

  display();

  insertAtEnd(40);

  insertAtEnd(50);

  insertAtEnd(60);

  display();

  insertAtPosition(35, 2);

  insertAtPosition(0, 5);

  insertAtPosition(65, 7);

  display();

  search(40);

  search(75);

  deleteAtBegin();

  deleteAtBegin();

  display();

  deleteAtEnd();

  deleteAtEnd();

  display();

  deleteAtPosition(4);

  deleteAtPosition(2);

  display();

  return 0;

}

void insertAtBegin(int x)

{

  struct node \*new;

  new = (struct node \*)malloc(sizeof(struct node));

  new->data = x;

  new->link = head;

  head = new;

  printf("\nInserted %d at the beginning", x);

  count++;

}

void insertAtEnd(int x)

{

  if (head == NULL)

    insertAtBegin(x);

  else

  {

    struct node \*temp = head;

    while (temp->link != NULL)

    {

      temp = temp->link;

    }

    struct node \*new;

    new = (struct node \*)malloc(sizeof(struct node));

    new->data = x;

    new->link = NULL;

    temp->link = new;

    printf("\nInserted %d at the end", x);

    count++;

  }

}

void insertAtPosition(int x, int pos)

{

  if (pos <= 0 || pos > (count + 1))

  {

    printf("Invalid position, insertion not possible");

    return;

  }

  if (pos == 1)

    insertAtBegin(x);

  else if (pos == (count + 1))

    insertAtEnd(x);

  else

  {

    struct node \*temp = head;

    for (int i = 1; i < (pos - 1); i++)

      temp = temp->link;

    struct node \*new;

    new = (struct node \*)malloc(sizeof(struct node));

    new->data = x;

    new->link = temp->link;

    temp->link = new;

    printf("\nInserted %d at position %d", x, pos);

    count++;

  }

}

void deleteAtBegin()

{

  if (head == NULL)

  {

    printf("\nLinked list is empty");

    return;

  }

  struct node \*temp = head;

  head = temp->link;

  printf("\nDeleted %d from the beginning", temp->data);

  free(temp);

  count--;

}

void deleteAtEnd()

{

  if (head == NULL)

  {

    printf("\nLinked list is empty");

    return;

  }

  if (head->link == NULL)

  {

    deleteAtBegin();

  }

  else

  {

    struct node \*prev = head;

    struct node \*temp = head->link;

    while (temp->link != NULL)

    {

      temp = temp->link;

      prev = prev->link;

    }

    prev->link = NULL;

    printf("\nDeleted %d from the end", temp->data);

    free(temp);

    count--;

  }

}

void deleteAtPosition(int pos)

{

  if (head == NULL)

  {

    printf("\nLinked list is empty");

    return;

  }

  if (pos <= 0 || pos > count)

  {

    printf("Invalid position, deletion not possible");

    return;

  }

  if (pos == 1)

    deleteAtBegin();

  else if (pos == count)

    deleteAtEnd();

  else

  {

    struct node \*prev = head;

    struct node \*temp = head->link;

    for (int i = 2; i < pos; i++)

    {

      temp = temp->link;

      prev = prev->link;

    }

    prev->link = temp->link;

    printf("\nDeleted %d from position %d", temp->data, pos);

    free(temp);

    count--;

  }

}

void display()

{

  if (head == NULL)

  {

    printf("\nNo elements in the linked list");

    return;

  }

  struct node \*temp = head;

  printf("\nElements in the linked list are: \n");

  printf("head");

  while (temp != NULL)

  {

    printf(" --> %d", temp->data);

    temp = temp->link;

  }

  printf("\n");

}

void search(int key)

{

  if (head == NULL)

  {

    printf("\nNo elements in the linked list");

    return;

  }

  struct node \*temp = head;

  int flag = 1, pos = 1;

  printf("\nSearching %d: ", key);

  while (temp != NULL)

  {

    if (temp->data == key)

    {

      printf("\nFound at position: %d", pos);

      flag = 0;

    }

    temp = temp->link;

    pos++;

  }

  if (flag)

    printf("Key not found");

  printf("\n");

}