INSERTION AT THE BRGINING

ALGORTHIM

INSERTION AT THE BEGINNING

Step 1:Create a new node with the data to be inserted.

Step 2:Set the next pointer to the new node to the current head of the list.

Step 3:Update head of the list to the new node.

INSERTION AT THE RANDOM POSITION

Step 1: Create a new node with data to be inserted.

Step 2:Transverse the list to find the node that comes immediately before the desired position.

Step 3:Set the next pointer of the new node to the next pointer to the node found in step 2.

Step 4:Set the next pointer of the node found in step 2 to the new node.

INSERTING AT THE END

Step 1:create the new node with the data to be inserted.

Step 2:Transverse the list to find the last node.

Step 3:Set the next pointer of the last node to the new node.

DELETING AT THE BEGINNING

Step 1:store the current head of the list in a temporary variable.

Step 2:Update the head of the list to the new node.

Step 3:Free the memory allocated to the new node.

DELETING AT THE RANDOM POSITION

Step 1:Transverse the list to find the node that comes immediately before the node to be deleted.

Step 2:store the next pointer of the node to be deleted in a temporary node.

Step 3:Set the pointer of the previous node to the next pointer of the node to be deleted.

Step 4:Free the memory allocated to the node to be deleted.

DELETING AT THE END

Step 1:Transverse the list to find second to the last node.

Step 2:Set the next pointer of the second to last node to null.

Step 3:Free the memory allocated to the last node.

PROGRAM

#include <stdio.h>

#include <stdlib.h>

// Node structure

struct Node {

int data;

struct Node \*next;

};

// Linked list structure

struct LinkedList {

struct Node \*head;

int size;

};

// Creates a new node

struct Node\* create\_node(int data) {

struct Node\* new\_node = (struct Node\*) malloc(sizeof(struct Node));

new\_node->data = data;

new\_node->next = NULL;

return new\_node;

}

// Initializes a new linked list

struct LinkedList\* init\_linked\_list() {

struct LinkedList\* linked\_list = (struct LinkedList\*) malloc(sizeof(struct LinkedList));

linked\_list->head = NULL;

linked\_list->size = 0;

return linked\_list;

}

// Adds a new node to the end of the linked list

void add\_to\_end(struct LinkedList\* linked\_list, int data) {

struct Node\* new\_node = create\_node(data);

if (linked\_list->head == NULL) {

linked\_list->head = new\_node;

} else {

struct Node\* current = linked\_list->head;

while (current->next != NULL) {

current = current->next;

}

current->next = new\_node;

}

linked\_list->size++;

}

// Adds a new node to the beginning of the linked list

void add\_to\_beginning(struct LinkedList\* linked\_list, int data) {

struct Node\* new\_node = create\_node(data);

new\_node->next = linked\_list->head;

linked\_list->head = new\_node;

linked\_list->size++;

}

// Removes the node with the given data from the linked list

void remove\_node(struct LinkedList\* linked\_list, int data) {

struct Node\* current = linked\_list->head;

struct Node\* previous = NULL;

while (current != NULL) {

if (current->data == data) {

if (previous == NULL) {

linked\_list->head = current->next;

} else {

previous->next = current->next;

}

linked\_list->size--;

free(current);

return;

}

previous = current;

current = current->next;

}

}

// Prints the linked list

void print\_linked\_list(struct LinkedList\* linked\_list) {

struct Node\* current = linked\_list->head;

while (current != NULL) {

printf("%d ", current->data);

current = current->next;

}

printf("\n");

}

int main() {

struct LinkedList\* linked\_list = init\_linked\_list();

add\_to\_end(linked\_list, 1);

add\_to\_end(linked\_list, 2);

add\_to\_end(linked\_list, 3);

add\_to\_end(linked\_list, 4);

print\_linked\_list(linked\_list); // Output: 1 2 3 4

add\_to\_beginning(linked\_list, 0);

print\_linked\_list(linked\_list); // Output: 0 1 2 3 4

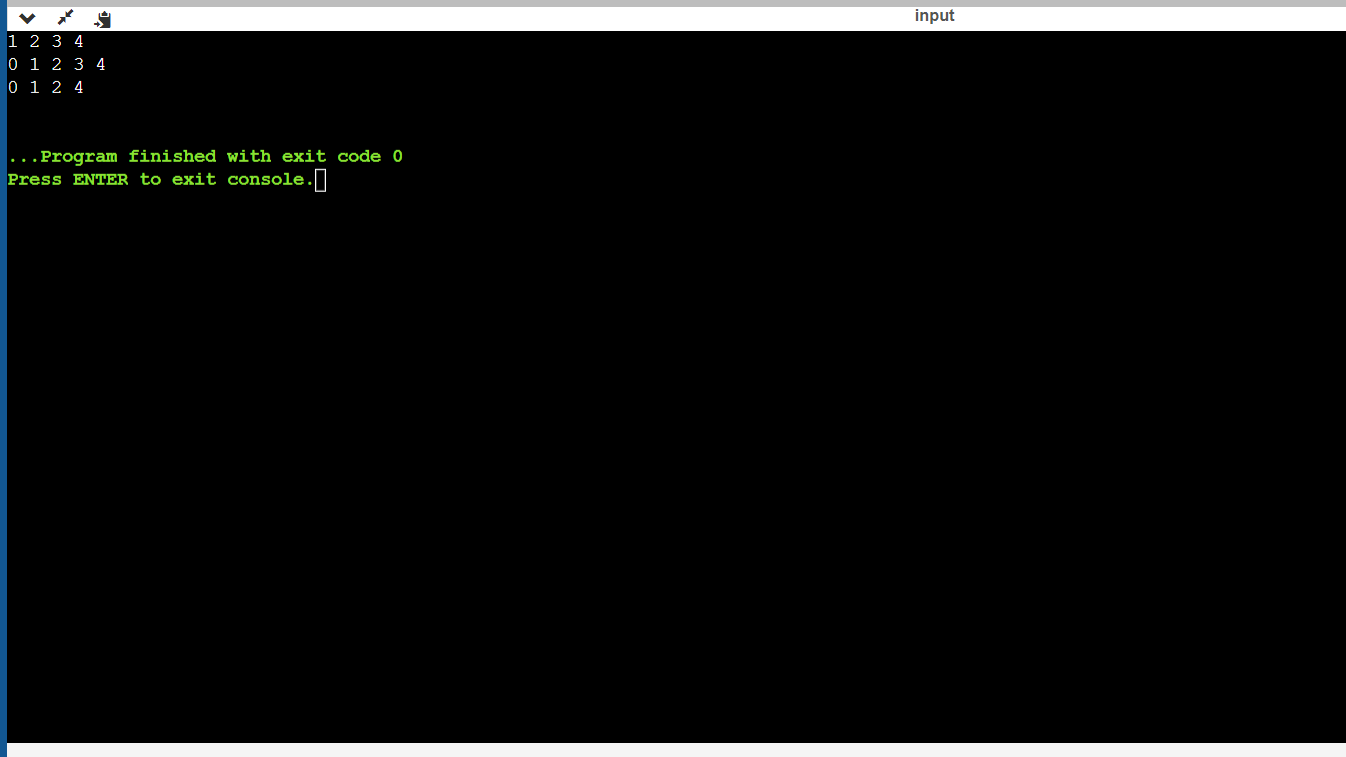
remove\_node(linked\_list, 3);

print\_linked\_list(linked\_list); // Output: 0 1 2 4

return 0;

}

OUTPUT



GITHUB LINK