**ASSIGNMENT- 3 DAY -3**

**Question 1**

**Write a function “insert\_any()” for inserting a node at any given position of the linked list. Assume**

**position starts at 0.**

**Question 2**

**Write a function “delete\_beg()” for deleting a node from the beginning of the linked list.**

ANS.)

#include <stdio.h>

#include <stdlib.h>

struct node{

int val;

struct node \*next;

};

/\*Delete the first node of a linked list.\*/

void delete\_beg(struct node \*\*head) {

struct node \*tmp;

/\*Linked list does not exist or the list is empty\*/

if(head == NULL || \*head == NULL) return;

/\*Storing the head to a temporary variable\*/

tmp = \*head;

/\*Moving head to the next node\*/

\*head = (\*head)->next;

/\*Deleting the first node\*/

free(tmp);

}

/\*Print the linked list\*/

void print\_list(struct node \*head) {

printf("H->");

while(head)

{

printf("%d->", head->val);

head = head->next;

}

printf("|||\n");

}

/\*Insert an element at the front of the list\*/

void insert\_front(struct node \*\*head, int value) {

struct node \* new\_node = NULL;

/\*Allocating memory for the new node\*/

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

if (new\_node == NULL)

{

printf("Failed to insert element. Out of memory");

}

new\_node->val = value;

/\*Pointing the new node to where head is currently pointing to\*/

new\_node->next = \*head;

/\*Pointing head to new node.\*/

\*head = new\_node;

}

void main()

{

int count = 0, i, val;

struct node \* head = NULL;

printf("Enter number of elements: ");

scanf("%d", &count);

for (i = 0; i < count; i++)

{

printf("Enter %dth element: ", i);

scanf("%d", &val);

insert\_front(&head, val);

}

printf("Initial Linked List: ");

print\_list(head);

delete\_beg(&head);

printf("Linked List after first node deleted: ");

print\_list(head);

}

**Question 3**

**Write a function “delete\_end()” for deleting a node from the end of the linked list.**

ANS.)

#include <stdio.h>

#include <stdlib.h>

/\* Structure of a node \*/

struct node {

int data; // Data

struct node \*next; // Address

}\*head;

void createList(int n);

void delete\_end();

void displayList();

int main()

{

int n, choice;

/\*

\* Create a singly linked list of n nodes

\*/

printf("Enter the total number of nodes: ");

scanf("%d", &n);

createList(n);

printf("\nData in the list \n");

displayList();

printf("\nPress 1 to delete last node: ");

scanf("%d", &choice);

/\* Delete last node from list \*/

if(choice == 1)

delete\_end();

printf("\nData in the list \n");

displayList();

return 0;

}

/\*

\* Create a list of n nodes

\*/

void createList(int n)

{

struct node \*newNode, \*temp;

int data, i;

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

/\*

\* If unable to allocate memory for head node

\*/

if(head == NULL)

{

printf("Unable to allocate memory.");

}

else

{

/\*

\* Input data of node from the user

\*/

printf("Enter the data of node 1: ");

scanf("%d", &data);

head->data = data; // Link the data field with data

head->next = NULL; // Link the address field to NULL

temp = head;

/\*

\* Create n nodes and adds to linked list

\*/

for(i=2; i<=n; i++)

{

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

/\* If memory is not allocated for newNode \*/

if(newNode == NULL)

{

printf("Unable to allocate memory.");

break;

}

else

{

printf("Enter the data of node %d: ", i);

scanf("%d", &data);

newNode->data = data; // Link the data field of newNode with data

newNode->next = NULL; // Link the address field of newNode with NULL

temp->next = newNode; // Link previous node i.e. temp to the newNode

temp = temp->next;

}

}

printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");

}

}

/\*

\* Delete last node of the linked list

\*/

void delete\_end()

{

struct node \*toDelete, \*secondLastNode;

if(head == NULL)

{

printf("List is already empty.");

}

else

{

toDelete = head;

secondLastNode = head;

/\* Traverse to the last node of the list \*/

while(toDelete->next != NULL)

{

secondLastNode = toDelete;

toDelete = toDelete->next;

}

if(toDelete == head)

{

head = NULL;

}

else

{

/\* Disconnect link of second last node with last node \*/

secondLastNode->next = NULL;

}

/\* Delete the last node \*/

free(toDelete);

printf("SUCCESSFULLY DELETED LAST NODE OF LIST\n");

}

}

/\*

\* Display entire list

\*/

void displayList()

{

struct node \*temp;

/\*

\* If the list is empty i.e. head = NULL

\*/

if(head == NULL)

{

printf("List is empty.");

}

else

{

temp = head;

while(temp != NULL)

{

printf("Data = %d\n", temp->data); // Print the data of current node

temp = temp->next; // Move to next node

}

}

}