**Assignment 3**

**Q1:-** write a function “insert\_any()” for inserting any node at any given position of the linked list. Assume position starts at 0.

**Answer :-**

#include <stdio.h>

#include <stdlib.h>

struct node {

int data;

struct node \*next;

}\*head;

void createList(int n);

void insertNodeAtMiddle(int data, int position);

void displayList();

int main()

{

int n, data, position;

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

scanf("%d", &n);

createList(n);

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

displayList();

printf("\nEnter data to insert at middle of the list: ");

scanf("%d", &data);

printf("Enter the position to insert new node: " );

scanf("%d", &position);

insertNodeAtMiddle(data, position);

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

displayList();

return 0;

}

void createList(int n)

{

struct node \*newNode, \*temp;

int data, i;

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

if(head == NULL)

{

printf("Unable to allocate memory.");

}

else

{

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

scanf("%d", &data);

head->data = data;

head->next = NULL;

temp = head;

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

{

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

if(newNode == NULL)

{

printf("Unable to allocate memory.");

break;

}

else

{

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

scanf("%d", &data);

newNode->data = data;

newNode->next = NULL;

temp->next = newNode;

temp = temp->next;

}

}

printf("\t LINKED LIST ARE CREATED SUCCESSFULLY\n");

}

}

void insertNodeAtMiddle(int data, int position)

{

int i;

struct node \*newNode, \*temp;

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

if(newNode == NULL)

{

printf("Unable to allocate memory.");

}

else

{

newNode->data = data;

newNode->next = NULL;

temp = head;

for(i=2; i<=position-1; i++)

{

temp = temp->next;

if(temp == NULL)

break;

}

if(temp != NULL)

{

newNode->next = temp->next;

temp->next = newNode;

printf("DATA INSERTED SUCCESSFULLY\n");

}

else

{

printf("UNABLE TO INSERT DATA AT THE GIVEN POSITION\n");

}

}

}

void displayList()

{

struct node \*temp;

if(head == NULL)

{

printf("List is empty.");

}

else

{

temp = head;

while(temp != NULL)

{

printf("Data = %d\n", temp->data);

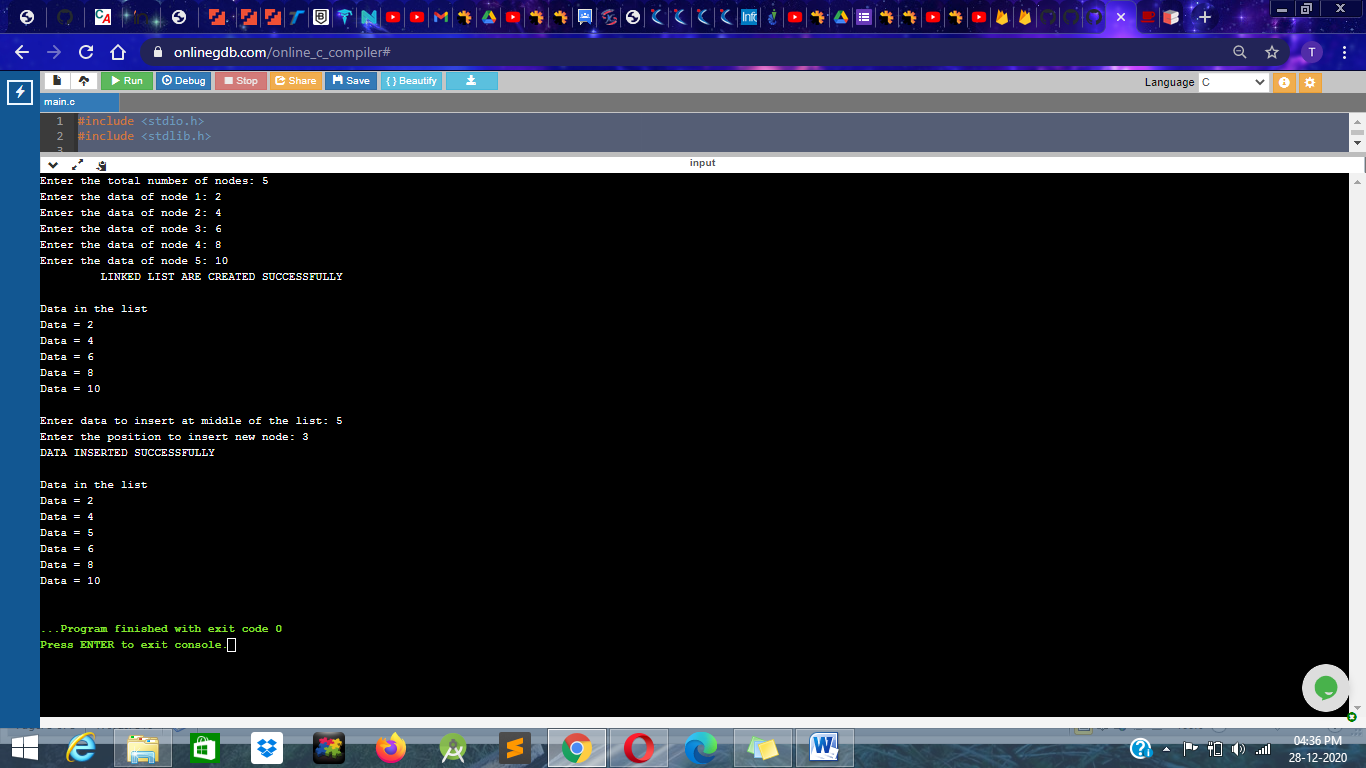
temp = temp->next;

}

}

}

**Output:-**



**Q2 :-- write a function “delete\_beg()” for deleting a node from the beginning of the linkedlist**

**Answer :-**

#include <stdio.h>

#include <stdlib.h>

struct node {

int data;

struct node \*next;

}\*head;

void createList(int n);

void deleteFirstNode();

void displayList();

int main()

{

int n, choice;

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

scanf("%d", &n);

createList(n);

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

displayList();

deleteFirstNode();

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

displayList();

return 0;

}

void createList(int n)

{

struct node \*newNode, \*temp;

int data, i;

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

if(head == NULL)

{

printf("Unable to allocate memory.");

}

else

{

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

scanf("%d", &data);

head->data = data;

head->next = NULL;

temp = head;

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

{

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

if(newNode == NULL)

{

printf("Unable to allocate memory.");

break;

}

else

{

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

scanf("%d", &data);

newNode->data = data;

newNode->next = NULL;

temp->next = newNode;

temp = temp->next;

}

}

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

}

}

void deleteFirstNode()

{

struct node \*toDelete;

if(head == NULL)

{

printf("List is already empty.");

}

else

{

toDelete = head;

head = head->next;

printf("\nDeleting first node from list = %d\n", toDelete->data);

free(toDelete);

printf("SUCCESSFULLY DELETED FIRST NODE FROM THE LIST\n");

}

}

void displayList()

{

struct node \*temp;

if(head == NULL)

{

printf("List is empty.");

}

else

{

temp = head;

while(temp != NULL)

{

printf("Data = %d\n", temp->data);

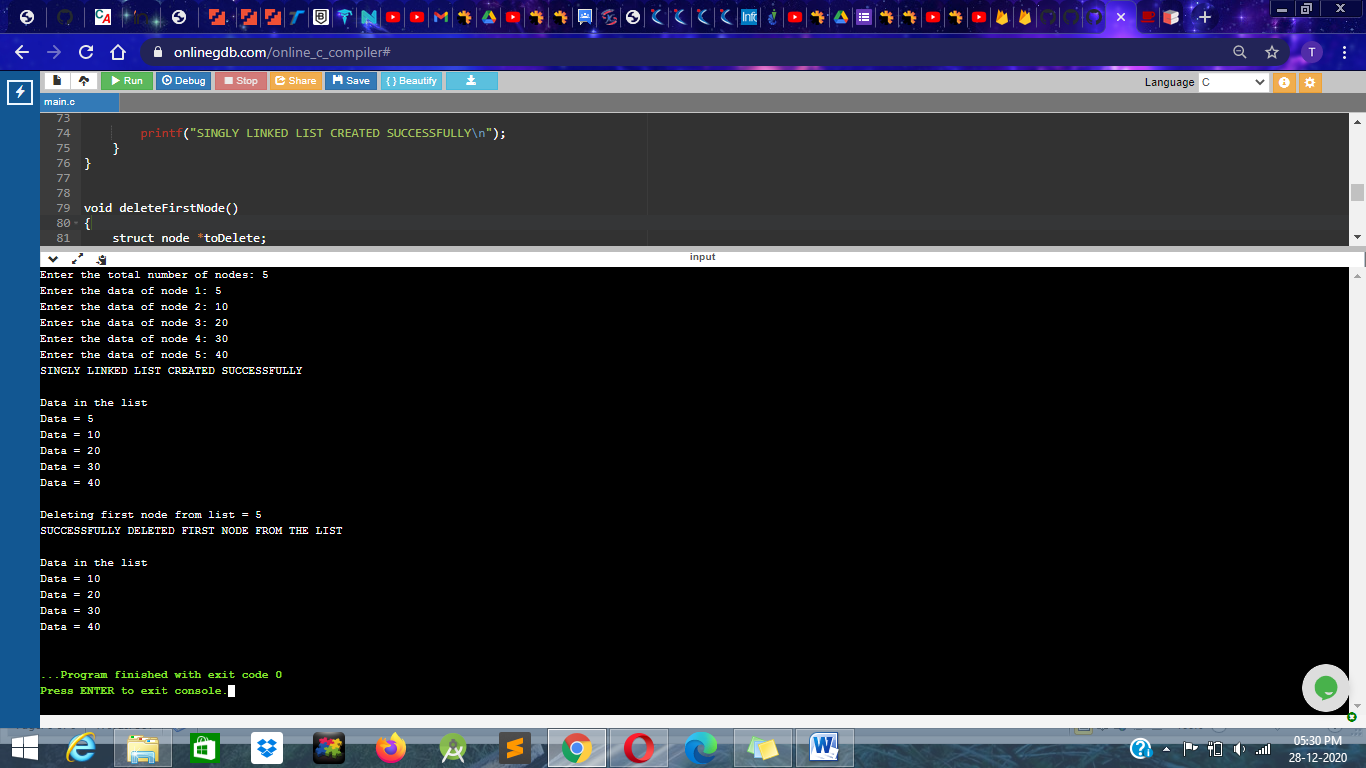
temp = temp->next;

}

}

}

**Output :-**



**Q3:--** write a function “delete\_end()” for deleting a node from the end of the linked list

**Answer:--**

#include <stdio.h>

#include <stdlib.h>

struct node {

int data;

struct node \*next;

}\*head;

void createList(int n);

void deleteLastNode();

void displayList();

int main()

{

int n, choice;

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

scanf("%d", &n);

createList(n);

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

displayList();

deleteLastNode();

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

displayList();

return 0;

}

void createList(int n)

{

struct node \*newNode, \*temp;

int data, i;

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

if(head == NULL)

{

printf("Unable to allocate memory.");

}

else

{

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

scanf("%d", &data);

head->data = data;

head->next = NULL;

temp = head;

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

{

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

if(newNode == NULL)

{

printf("Unable to allocate memory.");

break;

}

else

{

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

scanf("%d", &data);

newNode->data = data;

newNode->next = NULL;

temp->next = newNode;

temp = temp->next;

}

}

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

}

}

void deleteLastNode()

{

struct node \*toDelete, \*secondLastNode;

if(head == NULL)

{

printf("List is already empty.");

}

else

{

toDelete = head;

secondLastNode = head;

while(toDelete->next != NULL)

{

secondLastNode = toDelete;

toDelete = toDelete->next;

}

if(toDelete == head)

{

head = NULL;

}

else

{

secondLastNode->next = NULL;

}

free(toDelete);

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

}

}

void displayList()

{

struct node \*temp;

if(head == NULL)

{

printf("List is empty.");

}

else

{

temp = head;

while(temp != NULL)

{

printf("Data = %d\n", temp->data);

temp = temp->next;

}

}

}

**OutPut:-**

