**Lets Upgrade - Data Structures And Algorithms**

**Pratyusha Chakravarty**

**Assignment 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.

Ans:

#include<stdio.h>

#include<stdlib.h>

void insert\_any(int);

void create(int);

struct node

{

int data;

struct node \*next;

};

struct node \*head;

void main ()

{

printf("Lets Upgrade - Data Structures And Algorithms");

int choice,item,loc;

do

{

printf("\nEnter the item which you want to insert?\n");

scanf("%d",&item);

if(head == NULL)

{

create(item);

}

else

{

insert\_any(item);

}

printf("\nPress 0 to insert more ?\n");

scanf("%d",&choice);

}while(choice == 0);

}

void create(int item)

{

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

if(ptr == NULL)

{

printf("\nOVERFLOW ERROR\n");

}

else

{

ptr->data = item;

ptr->next = head;

head = ptr;

printf("\nSuccess! Node inserted\n");

}

}

void insert\_any(int item)

{

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

struct node \*temp;

int i,loc;

if(ptr == NULL)

{

printf("\nOVERFLOW ERROR");

}

else

{

printf("Enter the location where you want to insert");

scanf("%d",&loc);

ptr->data = item;

temp=head;

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

{

temp = temp->next;

if(temp == NULL)

{

printf("\ncan't insert\n");

return;

}

}

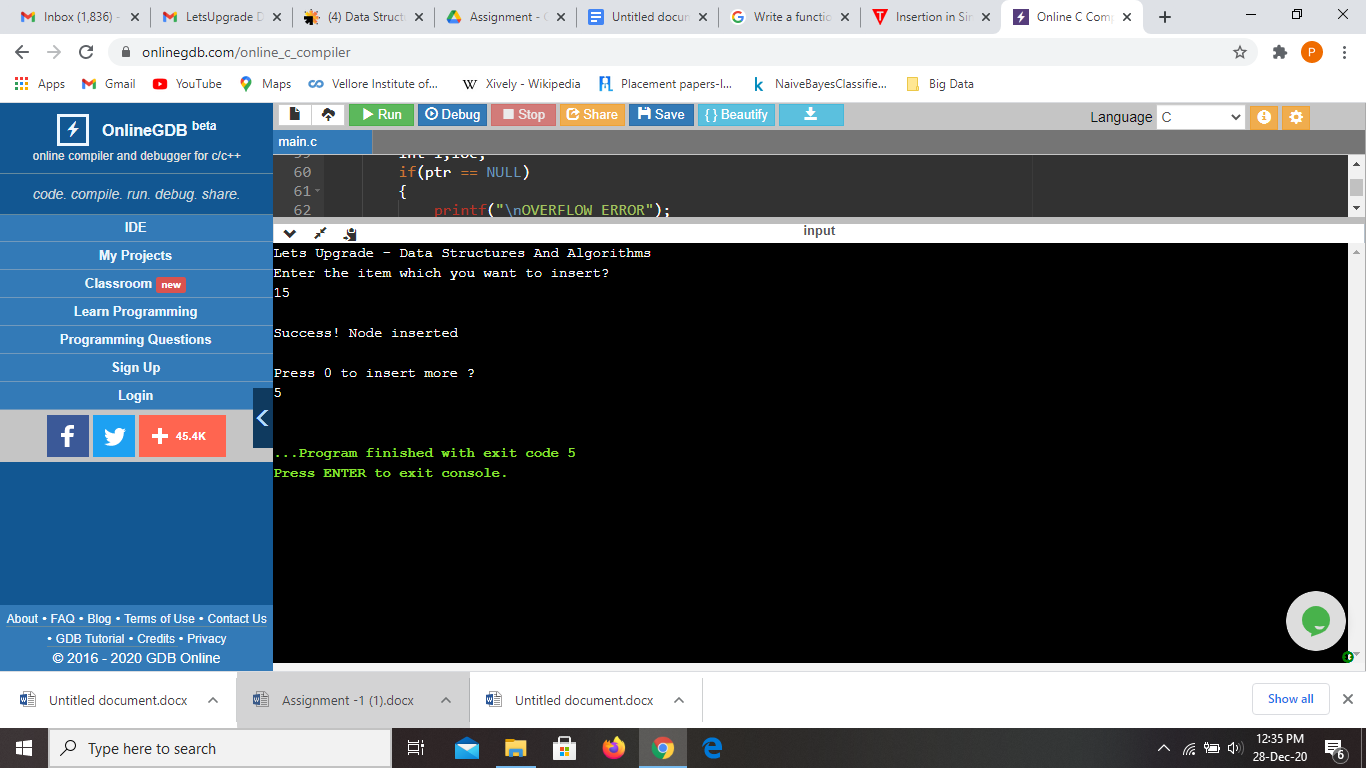
ptr ->next = temp ->next;

temp ->next = ptr;

printf("\nNode inserted");

}

}



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>

/\* Structure of a node \*/

struct node {

int data; // Data

struct node \*next; // Address

}\*head;

void createList(int n);

void delete\_beg();

void displayList();

int main()

{

printf("Lets Upgrade - Data Structures And Algorithms \n");

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 first node: ");

scanf("%d", &choice);

/\* Delete first node from list \*/

if(choice == 1)

delete\_beg();

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

{

/\*

\* In 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");

}

}

/\*

\* Deletes the first node of the linked list

\*/

void delete\_beg()

{

struct node \*toDelete;

if(head == NULL)

{

printf("List is already empty.");

}

else

{

toDelete = head;

head = head->next;

printf("\nData deleted = %d\n", toDelete->data);

/\* Clears the memory occupied by first node\*/

free(toDelete);

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

}

}

/\*

\* Displays the 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)

{

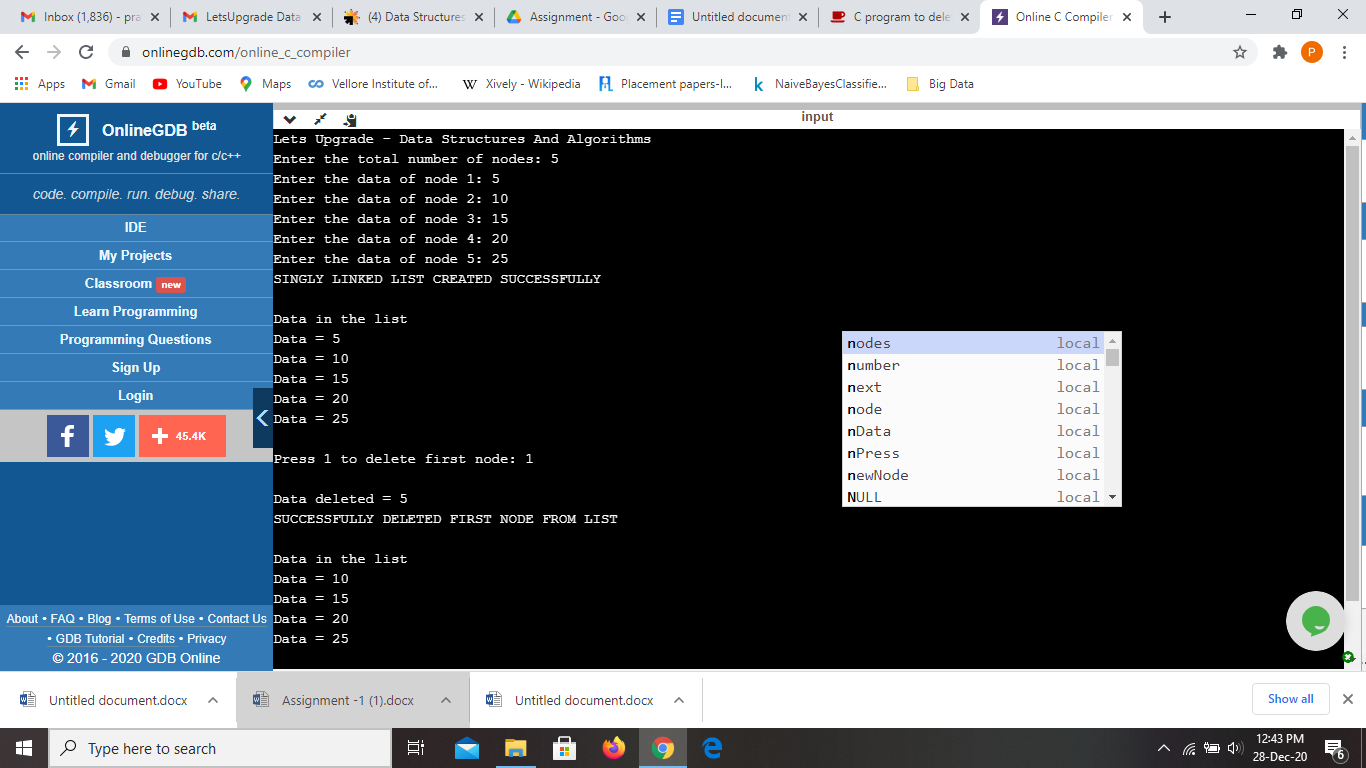
printf("Data = %d\n", temp->data); // Print data of current node

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

}

}

}



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>

struct node

{

int num; //Data of the node

struct node \*nextptr; //Address of the node

}\*stnode;

void createNodeList(int n); //function to create the list

void delete\_end(); //function to delete the last nodes

void displayList(); //function to display the list

int main()

{

printf("Lets Upgrade - Data Structures And Algorithms \n");

int n,num,pos;

printf("\n\n Linked List : Delete the last node of Singly Linked List :\n");

printf("---------------------------------------------------------------\n");

printf(" Input the number of nodes : ");

scanf("%d", &n);

createNodeList(n);

printf("\n Data entered in the list are : \n");

displayList();

delete\_end();

printf("\n The new list after deletion the last node are : \n");

displayList();

return 0;

}

void createNodeList(int n)

{

struct node \*fnNode, \*tmp;

int num, i;

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

if(stnode == NULL) //check whether the stnode is NULL and if so no memory allocation

{

printf(" Memory can not be allocated.");

}

else

{

// reads data for the node through keyboard

printf(" Input data for node 1 : ");

scanf("%d", &num);

stnode-> num = num;

stnode-> nextptr = NULL; //Links the address field to NULL

tmp = stnode;

//Creates n nodes and adds to linked list

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

{

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

if(fnNode == NULL) //check whether the fnnode is NULL and if so no memory allocation

{

printf(" Memory can not be allocated.");

break;

}

else

{

printf(" Input data for node %d : ", i);

scanf(" %d", &num);

fnNode->num = num; // links the num field of fnNode with num

fnNode->nextptr = NULL; // links the address field of fnNode with NULL

tmp->nextptr = fnNode; // links previous node i.e. tmp to the fnNode

tmp = tmp->nextptr;

}

}

}

}

// Deletes the last node of the linked list

void delete\_end()

{

struct node \*toDelLast, \*preNode;

if(stnode == NULL)

{

printf(" There is no element in the list.");

}

else

{

toDelLast = stnode;

preNode = stnode;

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

while(toDelLast->nextptr != NULL)

{

preNode = toDelLast;

toDelLast = toDelLast->nextptr;

}

if(toDelLast == stnode)

{

stnode = NULL;

}

else

{

/\* Disconnects the link of second last node with last node \*/

preNode->nextptr = NULL;

}

/\* Delete the last node \*/

free(toDelLast);

}

}

// function to display the entire list

void displayList()

{

struct node \*tmp;

if(stnode == NULL)

{

printf(" No data found in the empty list.");

}

else

{

tmp = stnode;

while(tmp != NULL)

{

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

tmp = tmp->nextptr; // advances the position of current node

}

}

}

