2 March, 2017

**Problems and Solutions from Data Structure Lab(Mid-term)**

**Problem 1: Find the factorial of a number using recursive function.**

#include<stdio.h>

void main()

{

int a, result;

printf("Enter an integer to find factorial: \n");

scanf("%d", &a);

result = factorial(a);

printf("Factorial of %d is: %d\n", a, result);

}

int factorial (int b)

{

if(b==0)

return 1;

else

return(b\*factorial(b-1));

}

**Problem 2: Find the Fibonacci series of number using recursive function.**

#include<stdio.h>

void main()

{

int a, b;

printf("Enter an integer to show it's Fibonacci Series:\n");

scanf("%d", &a);

printf("Fibonacci Series:");

for(b=0; b<a;b++)

{

printf("%d ", fibonacci(b));

}

}

int fibonacci(int a)

{

if(a==0||a==1)

{

return a;

}

else

{

return(fibonacci(a-1)+fibonacci(a-2));

}

}

**Problem 3: Find the power of a number using recursive function. (Base & power from user)**

#include<stdio.h>

int result(int b, int p)

{

if(p==0)

return 1;

else

b\*=result(b,p-1);

return b;

}

void main()

{

int base, power;

printf("Enter base & power: \n");

scanf("%d %d", &base, &power);

printf("Result: %d", result(base,power));

}

**Problem 4: Write a program using structure where “student” is a structure variable. And “s1” as “student” type variable. From the user, you have to collect First name, Last name and ID for one student. Print the collected data.**

#include<stdio.h>

struct student

{

char first[20];

char last[20];

int id;

}s1;

main()

{

printf("Enter first name:\n");

scanf("%s", s1.first);

printf("Enter last name:\n");

scanf("%s", s1.last);

printf("Enter id:\n");

scanf("%d", &s1.id);

printf("Name: %s %s\nId: %d\n\n", s1.first, s1.last, s1.id);

}

**Problem 5: Write a program using structure where “student” is a structure variable. And “s1” as “student” type variable. From the user you have to collect First name, Last name and ID for a number of students inputted by user. Print the collected data.**

#include<stdio.h>

struct student

{

char first[20];

char last[20];

int id;

};

main()

{

int a, b;

printf("Enter students number:\n");

scanf("%d", &b);

struct student s1[b];

for(a=0; a<b; a++)

{

printf("Enter first name:\n");

scanf("%s", s1[a].first);

printf("Enter last name:\n");

scanf("%s", s1[a].last);

printf("Enter id:\n");

scanf("%d", &s1[a].id);

}

for(a=0; a<b; a++)

{

printf("Name: %s %s\nId: %d\n\n", s1[a].first, s1[a].last, s1[a].id);

}

}

**Problem 6: Write a C program to find sum and average of ‘n’ elements entered by user. To perform this program, allocate memory dynamically using malloc() function.**

#include <stdio.h>

#include <stdlib.h>

int main()

{

int \*ptr, sum=0, i, n;

double ave;

printf("Enter how many numbers you want to add:\n");

scanf("%d",&n);

ptr = (int\*)malloc(n\*sizeof(int));

if(ptr==NULL)

{

printf("Not allocated\n");

exit (0);

}

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

{

scanf("%d",ptr+i);

sum+=\*(ptr+i);

}

ave = (double)sum/n;

printf("Sum = %d\nAverage = %0.2lf",sum,ave);

free(ptr);

return 0;

}

**Problem 7: Write a C program to find first ‘n’ odd numbers entered by user. To perform this program, allocate memory dynamically using malloc() function.**

#include <stdio.h>

#include <stdlib.h>

int main()

{

int \*p, n, i, j=1;

printf("Enter length of the series:\n");

scanf("%d",&n);

p = (int\*)malloc(n\*sizeof(int));

if(p==NULL)

{

printf("Not allocated\n");

exit (0);

}

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

{

\*(p+i) = j;

j+=2;

}

printf("Series :");

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

{

printf("%d ",\*(p+i));

}

free(p);

return 0;

}

**Problem 8: Write a C program to dynamically allocate memory for ‘n’ integers with malloc() function and print the addresses of the allocated memory units.**

#include <stdio.h>

#include <stdlib.h>

main()

{

int \*p, n, i;

printf("Enter the number of integers:\n");

scanf("%d",&n);

p = (int\*)malloc(n\*sizeof(int));

if(p==NULL)

{

printf("Not allocated\n");

exit (0);

}

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

printf("Address of unit %d is %p\n",i+1,p+i);

free(p);

return 0;

}

**Problem 9: Write a C program using malloc() to print a series for ‘n’ as given below;**

**When n = 3, Series: 4 12 20**

**When n = 4, Series: 4 12 20 28**

#include <stdio.h>

#include <stdlib.h>

main()

{

int \*ptr, n, i, j=4;

printf("Enter length of the series:\n");

scanf("%d",&n);

ptr = (int\*)calloc(n,sizeof(int));

if(ptr==NULL)

{

printf("Not allocated\n");

exit (0);

}

for(i=0;i<4;i++)

{

\*(ptr+i)=j;

j+=8;

}

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

{

printf("%d ",\*(ptr+i));

}

free(ptr);

return 0;

}

**Problem 10: Write a C program to input ‘n’ numbers using calloc() and linearly search a number from the input. If the searched number is in the list, print ”Found”, the memory address and the location unit for the given input. And if not found, simply print “Not Found”.**

#include <stdio.h>

#include <stdlib.h>

main()

{

int \*ptr, n, i, s, truth=0;

printf("Enter the number of input:\n");

scanf("%d",&n);

ptr = (int\*)calloc(n,sizeof(int));

if(ptr==NULL)

{

printf("Not allocated\n");

exit (0);

}

printf("Enter values:\n");

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

scanf("%d",&\*(ptr+i));

printf("Enter a digit to search: ");

scanf("%d",&s);

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

{

if(\*(ptr+i)==s)

{

truth++;

break;

}

}

if(truth==0)

printf("Not Found\n");

else

printf("Found at address %p, block %d\n",ptr+i,i+1);

}

**Problem 11: Write a C program to allocate memory dynamically for ‘n’ integers given by user using malloc() function and print the memory address of the allocated memory units. Now reallocate the memory again for different number of integers using realloc() function.**

#include <stdio.h>

#include <stdlib.h>

main()

{

int \*ptr, n, n2, n3, i;

printf("Enter the number of integers for which you want to allocate memory:\n");

scanf("%d",&n);

ptr = (int\*)malloc(n\*sizeof(int));

if(ptr==NULL)

{

printf("Not allocated\n");

exit (0);

}

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

{

printf("Address %p\n",ptr+i);

}

printf("Enter the new number of integers:\n");

scanf("%d",&n2);

ptr = realloc(ptr, n2);

for(i=0;i<n2;i++)

{

printf("Address %p\n",ptr+i);

}

printf("Enter the new new number of integers:\n");

scanf("%d",&n3);

ptr = realloc(ptr, n3);

for(i=0;i<n3;i++)

{

printf("Address %p\n",ptr+i);

}

return 0;

}

**Problem 12: Write a C program to create a structure named ‘node’ with members ‘int x’ and ‘int y’. Now create an object of mentioned structure in main function using malloc(). At last, assign some values to the structure members and print them out.**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int x;

int y;

};

main()

{

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

ptr->x=5;

ptr->y=6;

printf("%d %d",ptr->x,ptr->y);

free(ptr);

}

**Problem 13: Write a ‘C’ program to create a linked list for ‘n’ nodes inputted by user and print the list. You have to use two user-defined functions create\_ll() and display\_ll() to perform the mentioned jobs.**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

float data2;

struct node \*next;

}\*head=NULL, \*current = NULL;

void create\_ll()

{

int n,i,j;

float k;

printf("Enter how many node you want to create: ");

scanf("%d",&n);

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

{

struct node \*newnode;

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

if(newnode==NULL)

{

printf("Error\n");

exit;

}

printf("Enter data[%d] for new node:\n",i+1);

scanf("%d %f",&j,&k);

newnode->next = NULL;

newnode->data = j;

newnode->data2 = k;

if(head==NULL)

{

head = newnode;

current = newnode;

}

else

{

current->next= newnode;

current = newnode;

}

}

}

void display\_ll()

{

struct node \*position;

position = head;

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

while(position!=NULL)

{

printf("%d %0.2f-->\t",position->data,position->data2);

position = position->next;

}

printf("NULL\n");

}

main()

{

create\_ll();

display\_ll();

}

**Problem 14: Write a ‘C’ program to create a linked list and ask user every time if the user wants to input any more data in the list or not. Then print all the inputted data respectfully.**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

float data2;

struct node \*next;

}\*head=NULL, \*current = NULL;

void create\_ll()

{

int n,i,j;

float k;

char ch='y';

while(ch=='y')

{

struct node \*newnode;

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

if(newnode==NULL)

{

printf("Error\n");

exit;

}

printf("Enter data for new node:\n");

scanf("%d %f",&j,&k);

newnode->next = NULL;

newnode->data = j;

newnode->data2 = k;

if(head==NULL)

{

head = newnode;

current = newnode;

}

else

{

current->next= newnode;

current = newnode;

}

printf("Do you want to create another node? y/n\n");

ch=getch();

}

}

void display\_ll()

{

struct node \*position;

position = head;

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

while(position!=NULL)

{

printf("%d %0.2f-->\t",position->data,position->data2);

position = position->next;

}

printf("NULL\n");

}

main()

{

create\_ll();

display\_ll();

}

**Problem 15: Write a ‘C’ program to create a linked list for ‘n’ nodes inputted by user and print the list. Then you have to insert a node at the head of the linked list and print the updated list. You have to use three user-defined functions create\_ll(), insert\_first() and display\_ll() to perform the mentioned jobs.**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

float data2;

struct node \*next;

}\*head=NULL, \*current = NULL;

void insert\_first()

{

struct node \*newnode;

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

if(newnode==NULL)

{

printf("Error\n");

exit;

}

printf("Enter data for the nod:\n");

scanf("%d %f",&newnode->data,&newnode->data2);

newnode->next = NULL;

if(head==NULL)

{

head = newnode;

current = newnode;

}

else

{

newnode->next = head;

head = newnode;

}

}

void create\_ll()

{

int n,i,j;

float k;

printf("Enter how many node you want to create: ");

scanf("%d",&n);

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

{

struct node \*newnode;

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

if(newnode==NULL)

{

printf("Error\n");

exit;

}

printf("Enter data[%d] for new node:\n",i+1);

scanf("%d %f",&j,&k);

newnode->next = NULL;

newnode->data = j;

newnode->data2 = k;

if(head==NULL)

{

head = newnode;

current = newnode;

}

else

{

current->next= newnode;

current = newnode;

}

}

}

void display\_ll()

{

struct node \*position;

position = head;

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

while(position!=NULL)

{

printf("%d %0.2f-->\t",position->data,position->data2);

position = position->next;

}

printf("NULL\n");

}

main()

{

create\_ll();

display\_ll();

insert\_first();

display\_ll();

}

**Problem 16: Write a ‘C’ program to create a singly linked list and perform several operations like inserting nodes in different position, deleting a particular node, searching data in the list, counting nodes number in the list. You have to use different user defined functions to perform these operations.**

#include <stdio.h>

#include <stdlib.h>

struct l\_list

{

int a;

struct l\_list \*next;

};

typedef struct l\_list node;

node \*head=NULL, \*current=NULL;

int position\_s;

void sll\_create();

void sll\_display();

void sll\_finsert();

void sll\_linsert();

void sll\_minsert();

void sll\_del();

int sll\_search();

int sll\_count();

main()

{

int i=1, pos, key;;

printf("Choose option from below:\n");

while(i!=0)

{

printf("\n1: Create\n2: Insert\n3: Display\n4: Search\n5: Delete\n6: Count\n0: Exit\n");

scanf("%d",&i);

switch(i)

{

case 0:

printf("Goodbye!\n");

break;

case 1:

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

sll\_create();

break;

case 2:

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

printf("Choose insertion position: \n");

printf("1: First\n2: Last\n3: Mid\n");

int j;

printf("Your choice: ");

scanf("%d",&j);

switch(j)

{

case 1:

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

sll\_finsert();

break;

case 2:

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

sll\_linsert();

break;

case 3:

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

printf("Enter after which position you want to add node: \n");

sll\_minsert();

break;

}

break;

case 3:

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

sll\_display();

break;

case 4:

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

printf("Enter the value you want to search: ");

scanf("%d",&key);

pos = sll\_search(key);

if(position\_s!=0)

printf("\n%d found at position %d!\n",key, pos);

else

printf("\nValue not found!\n");

break;

case 5:

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

printf("Enter the key: ");

scanf("%d",&key);

pos = sll\_search(key);

sll\_del(pos);

break;

case 6:

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

printf("%d nodes found!\n",sll\_count());

break;

default:

printf("Invalid Input!\n");

break;

}

}

}

void sll\_create()

{

node \*newnode;

int i, j;

printf("How many nodes you want to create?\n");

scanf("%d",&j);

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

{

newnode = (node\*)malloc(sizeof(node));

if(newnode==NULL)

{

printf("Allocation ERROR!\n");

exit;

}

newnode->next = NULL;

printf("Enter data for node %d\n",i+1);

scanf("%d",&newnode->a);

if(head == NULL)

{

head = newnode;

current = newnode;

}

else

{

current->next = newnode;

current = newnode;

}

}

}

void sll\_finsert()

{

node \*newnode;

newnode = (node\*)malloc(sizeof(node));

if(newnode==NULL)

{

printf("Allocation ERROR!\n");

exit;

}

newnode->next = NULL;

printf("Insert data for new node:\n");

scanf("%d",&newnode->a);

if(head == NULL)

{

head = newnode;

current = newnode;

}

else

{

newnode->next = head;

head = newnode;

}

}

void sll\_linsert()

{

node \*newnode;

newnode = (node\*)malloc(sizeof(node));

if(newnode==NULL)

{

printf("Allocation ERROR!\n");

exit;

}

newnode->next = NULL;

printf("Insert data for new node:\n");

scanf("%d",&newnode->a);

if(head == NULL)

{

head = newnode;

current = newnode;

}

else

{

current->next = newnode;

current = newnode;

}

}

void sll\_minsert()

{

node \*newnode, \*temp, \*temp1;

int position, i;

newnode = (node\*)malloc(sizeof(node));

if(newnode==NULL)

{

printf("Allocation ERROR!\n");

exit;

}

scanf("%d",&position);

newnode->next = NULL;

printf("Insert data for new node:\n");

scanf("%d",&newnode->a);

if(head == NULL)

{

head = newnode;

current = newnode;

}

else

{

temp=head;

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

temp = temp->next;

temp1 = temp->next;

temp->next = newnode;

newnode->next=temp1;

}

}

void sll\_display()

{

node \*temp;

temp = head;

int i = 1;

while(temp!=NULL)

{

printf("%d-->\t",temp->a);

i++;

temp = temp->next;

}

printf("NULL\n");

}

int sll\_search(int x)

{

position\_s=0;

node \*temp;

int truth=0, i=1;

temp = head;

while(temp!=NULL)

{

if(temp->a==x)

{

truth++;

position\_s = i;

return position\_s;

}

i++;

temp = temp->next;

}

return position\_s;

}

void sll\_del(int d)

{

node \*temp, \*temp2;

int i;

temp = head;

if(d==1)

{

temp2=temp->next;

head = temp2;

}

else

{

for(i=1;i<d-1;i++)

temp = temp->next;

temp2 = temp->next;

temp->next = temp2->next;

if(temp2->next==NULL)

{

temp->next=NULL;

current = temp;

}

}

printf("Deleted successfully!\n");

}

int sll\_count()

{

node \*temp;

int c = 0;

temp=head;

while(temp!=NULL)

{

c++;

temp=temp->next;

}

return c;

}

**Problem 17: Write a ‘C’ program to perform different operation with doubly linked list. You have to create a linked list, then forwardly and backwardly display the list, after that forwardly and backwardly insert a node in the first position and finally you have to insert a node in mid position by both forward and backward access.**

#include <stdio.h>

#include <stdlib.h>

struct doubly

{

struct doubly \*prev;

int data;

struct doubly \*next;

};

typedef struct doubly node;

node \*head=NULL, \*tail=NULL;

void create();

void f\_display();

void b\_display();

void f\_insert\_first();

void f\_insert\_last();

void l\_insert\_first();

void l\_insert\_last();

void m\_insert\_first();

void m\_insert\_last();

void f\_delete();

void b\_delete();

int f\_search();

int b\_search();

int count();

int main()

{

int opt=1, opt2, truth, search=0;

printf("--->Choose option from below<---\n");

while(opt!=0)

{

printf("\n1. Create\n2. Insert\n3. Display\n4. Delete\n5. Search\n6. Count\n0. Exit\n");

printf("\nYour choice: ");

scanf("%d",&opt);

switch(opt)

{

case 1:

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

create();

break;

case 2:

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

printf("Choose insertion type:\n\n");

printf("1. First insert from first\n2. First insert from last\n3. Last insert from first\n4. Last insert from last\n5. Mid insert from first\n6. Mid insert from last\n\n");

printf("Your Choice: ");

scanf("%d",&opt2);

switch(opt2)

{

case 1:

printf("---First insert from first---\n\n");

f\_insert\_first();

break;

case 2:

printf("---First insert from last---\n\n");

f\_insert\_last();

break;

case 3:

printf("---Last insert from first---\n\n");

l\_insert\_first();

break;

case 4:

printf("---Last insert from last---\n\n");

l\_insert\_last();

break;

case 5:

printf("---Mid insert from first---\n\n");

m\_insert\_first();

break;

case 6:

printf("---Mid insert from last---\n\n");

m\_insert\_last();

break;

}

break;

case 3:

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

printf("Choose display type:\n\n");

printf("1. Forward display\n2. Backward display\n\n");

printf("Your choice: ");

scanf("%d",&opt2);

switch(opt2)

{

case 1:

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

f\_display();

break;

case 2:

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

b\_display();

break;

}

break;

case 4:

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

printf("Choose delete type:\n\n");

printf("1. Forward Delete\n2. Backward Delete\n\n");

printf("Your choice: ");

scanf("%d",&opt2);

switch(opt2)

{

case 1:

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

f\_delete();

break;

case 2:

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

b\_delete();

break;

}

printf("Node deleted successfully!\n");

break;

case 5:

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

printf("Choose Search type:\n\n");

printf("1. Forward Search\n2. Backward Search\n\n");

printf("Your Choice: ");

scanf("%d",&opt2);

switch(opt2)

{

case 1:

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

printf("Enter the value you want to search: ");

scanf("%d",&search);

truth=f\_search(search);

if(truth==0)

printf("Value does not exist!\n");

else

printf("Value found at position %d\n",truth);

break;

case 2:

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

printf("Enter the value you want to search: ");

scanf("%d",&search);

truth=b\_search(search);

if(truth==0)

printf("Value does not exist\n");

else

printf("Value found at position %d\n",truth);

break;

}

break;

case 6:

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

printf("%d nodes found in the list\n",count());

break;

}

}

}

void create()

{

node \*newnode;

int nodes\_num, i;

printf("How many nodes you want to create?\n\n");

printf("Enter here: ");

scanf("%d",&nodes\_num);

for(i=1;i<=nodes\_num;i++)

{

newnode = (node\*)malloc(sizeof(node));

if(newnode==NULL)

{

printf("Memory allocation failed!\n");

exit;

}

newnode->next=NULL;

newnode->prev=NULL;

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

scanf("%d",&newnode->data);

if(head==NULL)

{

head=newnode;

tail=newnode;

}

else

{

tail->next=newnode;

newnode->prev=tail;

tail=newnode;

}

}

}

void f\_display()

{

node \*temp;

temp=head;

printf("The List is:\n\n");

while(temp!=NULL)

{

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

temp=temp->next;

}

printf("NULL\n");

}

void b\_display()

{

node \*temp;

temp=tail;

printf("The List is:\n\n");

while(temp!=NULL)

{

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

temp=temp->prev;

}

printf("NULL\n");

}

void f\_insert\_first()

{

node \*newnode;

newnode = (node\*)malloc(sizeof(node));

if(newnode==NULL)

{

printf("Memory allocation failed!\n");

exit;

}

newnode->next=NULL;

newnode->prev=NULL;

if(head==NULL)

printf("You haven't created any node yet!\n");

else

{

printf("Insert the data for new node: ");

scanf("%d",&newnode->data);

newnode->next=head;

head->prev=newnode;

head=newnode;

}

}

void f\_insert\_last()

{

node \*newnode, \*current;

newnode = (node\*)malloc(sizeof(node));

if(newnode==NULL)

{

printf("Memory allocation failed!\n");

exit;

}

newnode->next=NULL;

newnode->prev=NULL;

current = tail;

while(current->prev!=NULL)

current=current->prev;

if(head==NULL)

printf("You haven't created any node yet!\n");

else

{

printf("Insert the data for new node: ");

scanf("%d",&newnode->data);

current->prev=newnode;

newnode->next=current;

head=newnode;

}

}

void l\_insert\_first()

{

node \*newnode, \*current;

newnode = (node\*)malloc(sizeof(node));

if(newnode==NULL)

{

printf("Memory allocation failed!\n");

exit;

}

newnode->next=NULL;

newnode->prev=NULL;

current = head;

while(current->next!=NULL)

current=current->next;

if(head==NULL)

printf("You haven't created any node yet!\n");

else

{

printf("Insert the data for new node: ");

scanf("%d",&newnode->data);

current->next=newnode;

newnode->prev=current;

tail=newnode;

}

}

void l\_insert\_last()

{

node \*newnode;

newnode = (node\*)malloc(sizeof(node));

if(newnode==NULL)

{

printf("Memory allocation failed!\n");

exit;

}

newnode->next=NULL;

newnode->prev=NULL;

if(head==NULL)

printf("You haven't created any node yet!\n");

else

{

printf("Insert the data for new node: ");

scanf("%d",&newnode->data);

tail->next=newnode;

newnode->prev=tail;

tail=newnode;

}

}

void m\_insert\_first()

{

int i, j;

node \*temp, \*temp1, \*newnode;

newnode = (node\*)malloc(sizeof(node));

if(newnode==NULL)

{

printf("Memory allocation failed!\n");

exit;

}

newnode->next=NULL;

newnode->prev=NULL;

if(head==NULL)

printf("You haven't created any node yet!\n");

else

{

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

scanf("%d",&j);

if(j==1)

f\_insert\_first();

else if(j==count())

l\_insert\_first();

else

{

printf("Insert the data for new node: ");

scanf("%d",&newnode->data);

temp = head;

for(i=1;i<j-1;i++)

temp=temp->next;

temp1 = temp->next;

temp->next=newnode;

newnode->next=temp1;

newnode->prev=temp;

temp1->prev=newnode;

}

}

}

m\_insert\_last()

{

int i, j;

node \*temp, \*temp1, \*newnode;

newnode = (node\*)malloc(sizeof(node));

if(newnode==NULL)

{

printf("Memory allocation failed!\n");

exit;

}

newnode->next=NULL;

newnode->prev=NULL;

if(head==NULL)

printf("You haven't created any node yet!\n");

else

{

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

scanf("%d",&j);

if(j==1)

f\_insert\_first();

else if(j==count())

l\_insert\_first();

else

{

printf("Insert the data for new node: ");

scanf("%d",&newnode->data);

temp = tail;

int counter=count()-j;

printf("\n\nTest: %d\n\n",counter);

for(i=1;i<counter+1;i++)

temp=temp->prev;

temp1=temp->prev;

temp1->next=newnode;

newnode->prev = temp1;

newnode->next = temp;

temp->prev = newnode;

}

}

}

void f\_delete()

{

node \*temp, \*temp1;

int pos, i;

printf("Enter the position you want to delete: ");

scanf("%d",&pos);

if(pos==1)

{

head=head->next;

head->prev=NULL;

}

else if(pos==count())

{

temp = head;

while(temp->next!=NULL)

temp=temp->next;

tail = temp->prev;

tail->next=NULL;

}

else

{

temp = head;

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

temp = temp->next;

temp1=temp->next;

temp->next=temp1->next;

temp1->prev=temp->prev;

}

}

void b\_delete()

{

node \*temp, \*temp1;

int pos, i;

printf("Enter the position you want to delete: ");

scanf("%d",&pos);

if(pos==1)

{

temp = tail;

while(temp->prev!=NULL)

temp=temp->prev;

head = temp->next;

head->prev=NULL;

}

else if(pos==count())

{

tail=tail->prev;

tail->next=NULL;

}

else

{

temp = tail;

pos = count()-pos;

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

temp = temp->prev;

temp1=temp->prev;

temp->prev=temp1->prev;

temp1->next=temp->next;

}

}

int f\_search(int value)

{

int pos=1;

node \*temp = head;

while(temp!=NULL)

{

if(temp->data==value)

return pos;

pos++;

temp=temp->next;

}

return 0;

}

int b\_search(int value)

{

int pos=0;

node \*temp = tail;

while(temp!=NULL)

{

if(temp->data==value)

return (count()-pos);

pos++;

temp=temp->prev;

}

}

int count()

{

node \*temp;

int counter=1;

temp=head;

while(temp->next!=NULL)

{

temp=temp->next;

counter++;

}

return counter;

}

**Problem 18: Write a ‘C’ program to create a stack(size given by user), then push, pop, and display elements as long as user wants.**

#include <stdio.h>

#include <stdlib.h>

int top = -1, stk[100], size;

void push(int value)

{

if(top==size-1)

printf("Stack Overflow!\n");

else

stk[++top]=value;

}

int pop()

{

if(top==-1)

printf("Stack Underflow!\n");

else

return stk[top--];

}

void display()

{

int i;

for(i=top;i>=0;i--)

printf("%d\n",stk[i]);

}

int main()

{

int item, check, loop=1;

printf("Enter stack size within 100: ");

scanf("%d",&size);

printf("\n1. Push\n2. Pop\n3. Display\n4. Break\n");

while(loop)

{

int option;

printf("\nYour choice: ");

scanf("%d",&option);

switch(option)

{

case 1:

printf("Enter value to push: ");

scanf("%d",&item);

push(item);

break;

case 2:

check=pop();

if(check==0)

break;

else

printf("%d popped successfully!\n",check);

break;

case 3:

printf("\nDisplay\n\n");

display();

break;

case 4:

printf("Goodbye!\n");

loop=0;

break;

}

}

return 0;

}

-End-