**DSA PRACTICAL FILE**

(CodeChef and Online GDB C language IDE is used for all the programs)

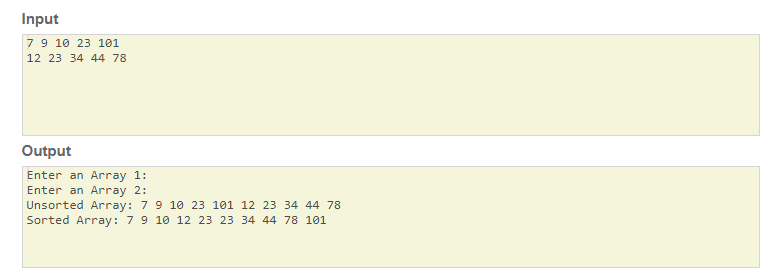
Practical File programs from 5 to 8

Name: Hardik Chhabra

ECAM-1

Roll no.: 2020UEA6504

**OUTPUT-5**



**Practical-5**

**Q. Write a program to merge to sorted arrays.**

#include<stdio.h>

void mergeSort(int[],int,int);

void merge(int[],int,int,int);

void main ()

{

int a[5],b[5],c[10];//= {10, 9, 7, 101, 23, 44, 12, 78, 34, 23};

int i;

printf("Enter an Array 1: ");

for (int i = 0; i < 5; i++) {

scanf("%d",&a[i]);

}

printf("\nEnter an Array 2: ");

for (int i = 0; i < 5; i++) {

scanf("%d",&b[i]);

}

//Merging

for (int i = 0; i < 10; i++) {

if(i<=4) {c[i]=a[i];}

else {

int j=i-5;

c[i]=b[j];

}

}

printf("\nUnsorted Array:");

for (int i = 0; i < 10; i++) {

printf(" %d",c[i]);

}

mergeSort(c,0,9);

printf("\nSorted Array:");

for(i=0;i<10;i++)

{

printf(" %d",c[i]);

}

}

void mergeSort(int a[], int beg, int end)

{

int mid;

if(beg<end)

{

mid = (beg+end)/2;

mergeSort(a,beg,mid);

mergeSort(a,mid+1,end);

merge(a,beg,mid,end);

}

}

void merge(int a[], int beg, int mid, int end)

{

int i=beg,j=mid+1,k,index = beg;

int temp[10];

while(i<=mid && j<=end)

{

if(a[i]<a[j])

{

temp[index] = a[i];

i = i+1;

}

else

{

temp[index] = a[j];

j = j+1;

}

index++;

}

if(i>mid)

{

while(j<=end)

{

temp[index] = a[j];

index++;

j++;

}

}

else

{

while(i<=mid)

{

temp[index] = a[i];

index++;

i++;

}

}

k = beg;

while(k<index)

{

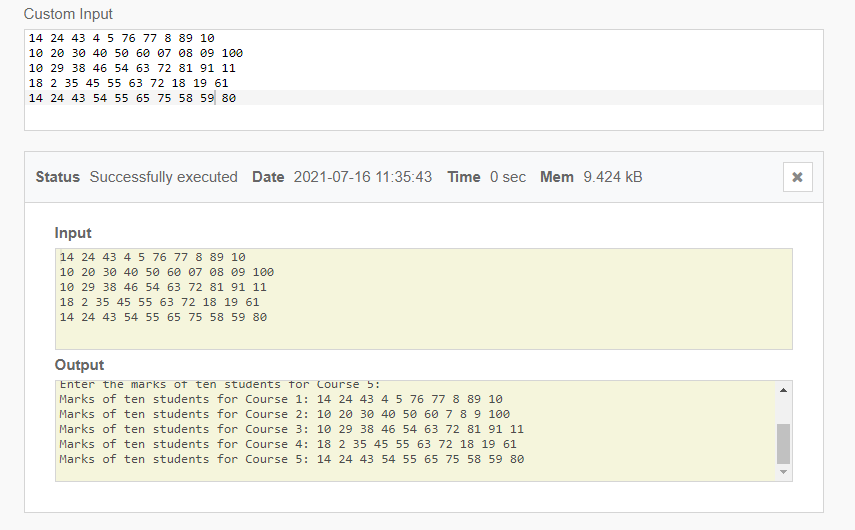
a[k]=temp[k];

k++;

}

}

**OUTPUT-6**

**Practical-6**

**Q. Write a program to store the marks obtained by 10 students in 5 courses in a two dimensional array.**

#include <stdio.h>

int main(void) {

// your code goes here

int m[5][10];

for (int i=0;i<5;i++) {

printf("Enter the marks of ten students for Course %d: ",i+1);

for(int j=0;j<10;j++) {

scanf("%d",&m[i][j]);

}

printf("\n");

}

for (int i=0;i<5;i++) {

printf("Marks of ten students for Course %d:",i+1);

for(int j=0;j<10;j++) {

printf(" %d",m[i][j]);

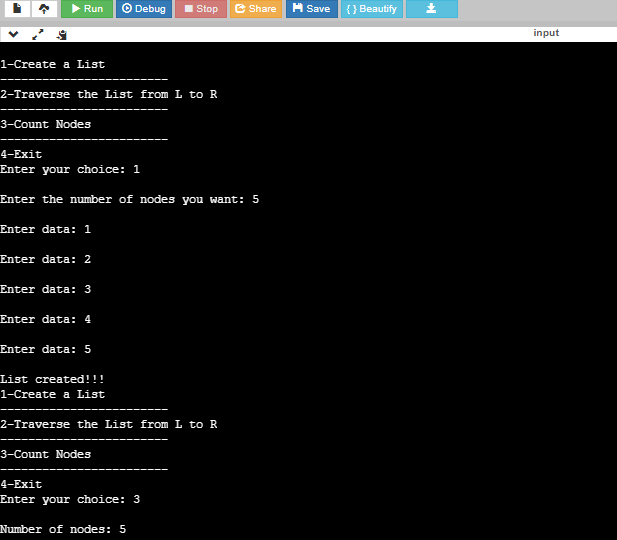
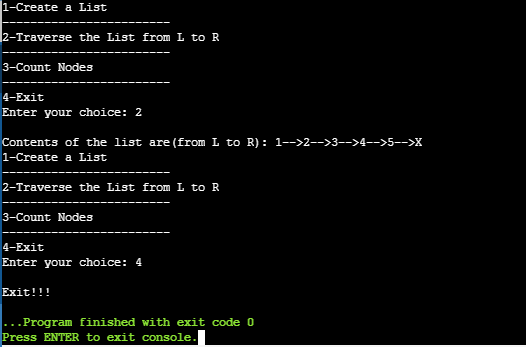
}

printf("\n");

}

return 0;

}

**OUTPUT-7**

**Practical-7**

**Q. Write a program to implement a linked list.**

#include<stdio.h>

#include<stdlib.h>

struct LL {

int data;

struct LL \*next;

};

typedef struct LL node;

node \*start=NULL;

node\* getnode() {

node\* newnode;

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

printf("\nEnter data: ");

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

newnode->next=NULL;

return newnode;

}

int menu() {

int ch;

printf("\n1-Create a List");

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

printf("\n2-Traverse the List from L to R");

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

printf("\n3-Count Nodes");

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

printf("\n4-Exit");

printf("\nEnter your choice: ");

scanf("%d",&ch);

return ch;

}

void createLL(int n) {

node \*temp, \*newnode;

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

newnode=getnode();

if(start==NULL) {start=newnode;}

else {

temp=start;

while(temp->next!=NULL) {temp=temp->next;}

temp->next=newnode;

}

}

}

int countnode(node \*ptr) {

int ctr=0;

while(ptr!=NULL) {

ctr++;

ptr=ptr->next;

}

return ctr;

}

void traverse() {

node \*temp;

temp=start;

if(start==NULL) {printf("\nList is Empty!!!");}

else {

printf("\nContents of the list are(from L to R): ");

while(temp!=NULL) {

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

temp=temp->next;

}

}

printf("X");

}

int main(void) {

int ch,n;

while(1) {

ch=menu();

switch(ch) {

case 1: if(start==NULL) {

printf("\nEnter the number of nodes you want: ");

scanf("%d",&n);

createLL(n);

printf("\nList created!!!");

}else {printf("List already exists!!!");}

break;

case 2: traverse();

break;

case 3: printf("\nNumber of nodes: %d",countnode(start));

break;

case 4: printf("\nExit!!!");

exit(0);

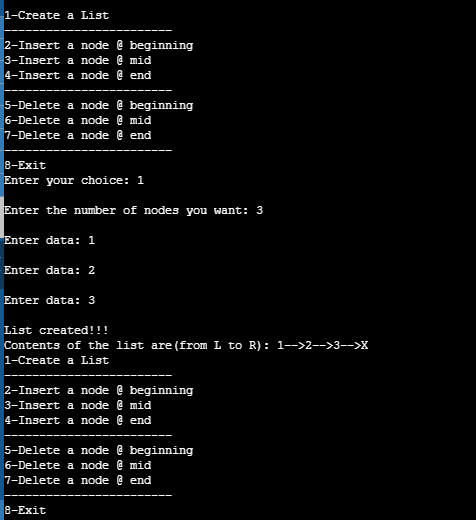
}

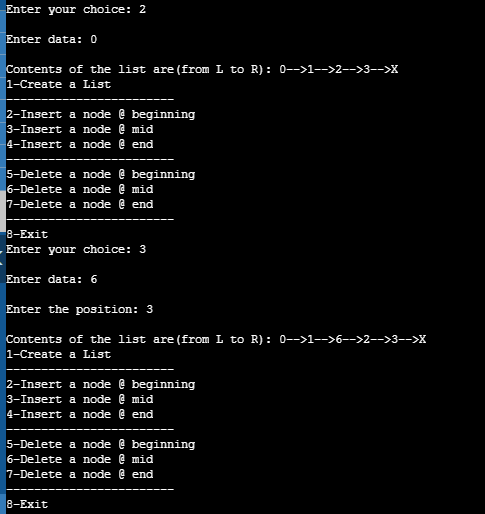
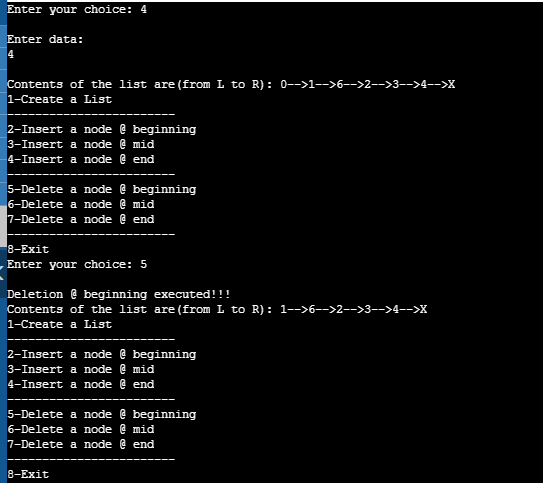
}

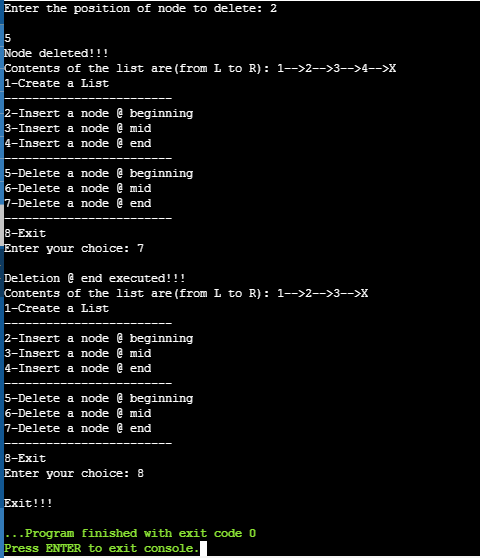
return 0;

}

**OUTPUT-8**







**Practical-8**

**Q. Write a program to insert a node from linked list and delete a node from linked list.**

#include<stdio.h>

#include<stdlib.h>

struct LL {

int data;

struct LL \*next;

};

typedef struct LL node;

node \*start=NULL;

node\* getnode() {

node\* newnode;

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

printf("\nEnter data: ");

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

newnode->next=NULL;

return newnode;

}

int menu() {

int ch;

printf("\n1-Create a List");

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

printf("\n2-Insert a node @ beginning");

printf("\n3-Insert a node @ mid");

printf("\n4-Insert a node @ end");

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

printf("\n5-Delete a node @ beginning");

printf("\n6-Delete a node @ mid");

printf("\n7-Delete a node @ end");

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

printf("\n8-Exit");

printf("\nEnter your choice: ");

scanf("%d",&ch);

return ch;

}

void createLL(int n) {

node \*temp, \*newnode;

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

newnode=getnode();

if(start==NULL) {start=newnode;}

else {

temp=start;

while(temp->next!=NULL) {temp=temp->next;}

temp->next=newnode;

}

}

}

int countnode(node \*ptr) {

int ctr=0;

while(ptr!=NULL) {

ctr++;

ptr=ptr->next;

}

return ctr;

}

void traverse() {

node \*temp;

temp=start;

if(start==NULL) {printf("\nList is Empty!!!");}

else {

printf("\nContents of the list are(from L to R): ");

while(temp!=NULL) {

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

temp=temp->next;

}

printf("X");

}

}

void ins\_at\_beg() {

node \*newnode;

newnode=getnode();

if(start==NULL) {start=newnode;}

else {

newnode->next=start;

start=newnode;

}

traverse();

}

void ins\_at\_end() {

node \*newnode,\*temp;

newnode=getnode();

if(start==NULL) {start=newnode;}

else {

temp=start;

while(temp->next!=NULL) {temp=temp->next;}

temp->next=newnode;

}

traverse();

}

void ins\_at\_mid() {

node \*newnode, \*temp, \*prev;

int pos,nodectr,ctr=1;

newnode=getnode();

printf("\nEnter the position: ");

scanf("%d",&pos);

nodectr=countnode(start);

if(pos>1 && pos<nodectr) {

temp=prev=start;

while(ctr<pos) {

prev=temp;

temp=temp->next;

ctr++;

}

prev->next=newnode;

newnode->next=temp;

}

traverse();

}

void del\_at\_beg() {

node \*temp;

if(start==NULL) {

printf("\nNo nodes exist!!!");

return;

}else {

temp=start;

start=temp->next;

free(temp);

printf("\nDeletion @ beginning executed!!!");

}

traverse();

}

void del\_at\_end() {

node \*temp, \*prev;

if(start==NULL) {

printf("\nNo nodes exist!!!");

return;

}else {

temp=start;

prev=start;

while(temp->next!=NULL) {

prev=temp;

temp=temp->next;

}

prev->next=NULL;

free(temp);

printf("\nDeletion @ end executed!!!");

}

traverse();

}

void del\_at\_mid() {

int ctr=1,pos,nodectr;

node \*temp,\*prev;

if(start==NULL) {

printf("Empty List!!!");

return;

}else {

printf("\nEnter the position of node to delete: ");

scanf("%d",&pos);

nodectr=countnode(start);

if(pos>1 && pos<nodectr) {

temp=prev=start;

while(ctr<pos) {

prev=temp;

temp=temp->next;

ctr++;

}

prev->next=temp->next;

free(temp);

printf("\nNode deleted!!!");

}else {printf("\nInvalid position!!!");}

}

traverse();

}

int main(void) {

int ch,n;

while(1) {

ch=menu();

switch(ch) {

case 1: if(start==NULL) {

printf("\nEnter the number of nodes you want: ");

scanf("%d",&n);

createLL(n);

printf("\nList created!!!");

traverse();

}else {

printf("List already exists!!!");

traverse();

}

break;

case 2: ins\_at\_beg();

break;

case 3: ins\_at\_mid();

break;

case 4: ins\_at\_end();

break;

case 5: del\_at\_beg();

break;

case 6: del\_at\_mid();

break;

case 7: del\_at\_end();

break;

case 8: printf("\nExit!!!");

exit(0);

}

}

return 0;

}