**R.Harini**

**18BCE1010**

**1)**

#include<stdio.h>

#include<stdlib.h>

struct Node{

int data;

struct Node\* next;

};

void display(struct Node\* head){

struct Node\* temp=(struct Node\*)malloc(sizeof(struct Node));

temp=head->next;

printf("Displaying the linked list\n");

while(temp!=NULL){

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

temp=temp->next;

}

}

void add(struct Node\* head, int i){

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

newnode->data=i;

newnode->next=NULL;

if (head->next==NULL){

head->next=newnode;

return;

}

struct Node\* temp=(struct Node\*)malloc(sizeof(struct Node));

temp=head->next;

while (temp->next!=NULL){

temp=temp->next;

}

temp->next=newnode;

}

void insert(struct Node\* head,int a, int i){

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

newnode->data=i;

struct Node\* temp=(struct Node\*)malloc(sizeof(struct Node));

temp=head->next;

while (temp!=NULL){

if (temp->data==a){

newnode->next=temp->next;

temp->next=newnode;

return;

}

else{

temp=temp->next;

}

}

printf("Node not found\n");

}

void find(struct Node\* head,int i){

struct Node\* temp=(struct Node\*)malloc(sizeof(struct Node));

temp=head->next;

while(temp!=NULL){

if(temp->data==i){

printf("%d found\n",i);

return;

}

else{

temp=temp->next;

}

}

printf("Not found\n");

}

int o=0,e=0;

void odd\_even(struct Node\* head){

struct Node\* temp=(struct Node\*)malloc(sizeof(struct Node));

temp=head->next;

while(temp!=NULL){

if (temp->data%2==0){

e++;

}

else{

o++;

}

temp=temp->next;

}

}

struct Node\* deletenode(struct Node\* head,int i){

struct Node\* del=(struct Node\*)malloc(sizeof(struct Node));

if (head->next->data==i){

del=head->next;

head->next=del->next;

return del;

}

struct Node\* temp=(struct Node\*)malloc(sizeof(struct Node));

temp=head->next;

while(temp->next!=NULL){

if (temp->next->data==i){

del=temp->next;

temp->next=del->next;

return del;

}

temp=temp->next;

}

}

void remove\_odd(struct Node\* head, struct Node\* head1){

struct Node\* temp=(struct Node\*)malloc(sizeof(struct Node));

temp=head->next;

struct Node\* n=(struct Node\*)malloc(sizeof(struct Node));

while (temp!=NULL){

if ((temp->data)%2==1){

n=deletenode(head, temp->data);

add(head1,n->data);

}

temp=temp->next;

}

}

int main(){

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

head->next=NULL;

struct Node\* head1=(struct Node\*)malloc(sizeof(struct Node));

head1->next=NULL;

int c=1,n,a;

int i;

while (c>0){

printf("\*\*\*\*MENU\*\*\*\*\*\n");

printf("1->Add a number\n2->Display the list\n3->Insert a number\n");

printf("4->Find a number\n5->Check how many odd or even numbers\n");

printf("6->Delete a number\n7->Remove all odd nos\n0->End the process\n\*\*\*\*\*\*\n");

scanf("%d",&c);

switch (c) {

case 1:

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

scanf("%d",&n);

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

printf("Enter the value:");

scanf("%d",&a);

add(head,a);

}

break;

case 2:

display(head);

break;

case 3:

printf("Enter the number to be inserted:");

scanf("%d",&a);

printf("Enter the number after which to be inserted:");

scanf("%d",&i);

insert(head,i,a);

break;

case 4:

printf("Enter the number to be found:");

scanf("%d",&a);

find(head,a);

break;

case 5:

odd\_even(head);

printf("Odd nos:%d\n Even nos:%d\n",o,e);

break;

case 6:

printf("Enter the number to be deleted:");

scanf("%d",&a);

deletenode(head,a);

break;

case 7:

remove\_odd(head,head1);

display(head1);

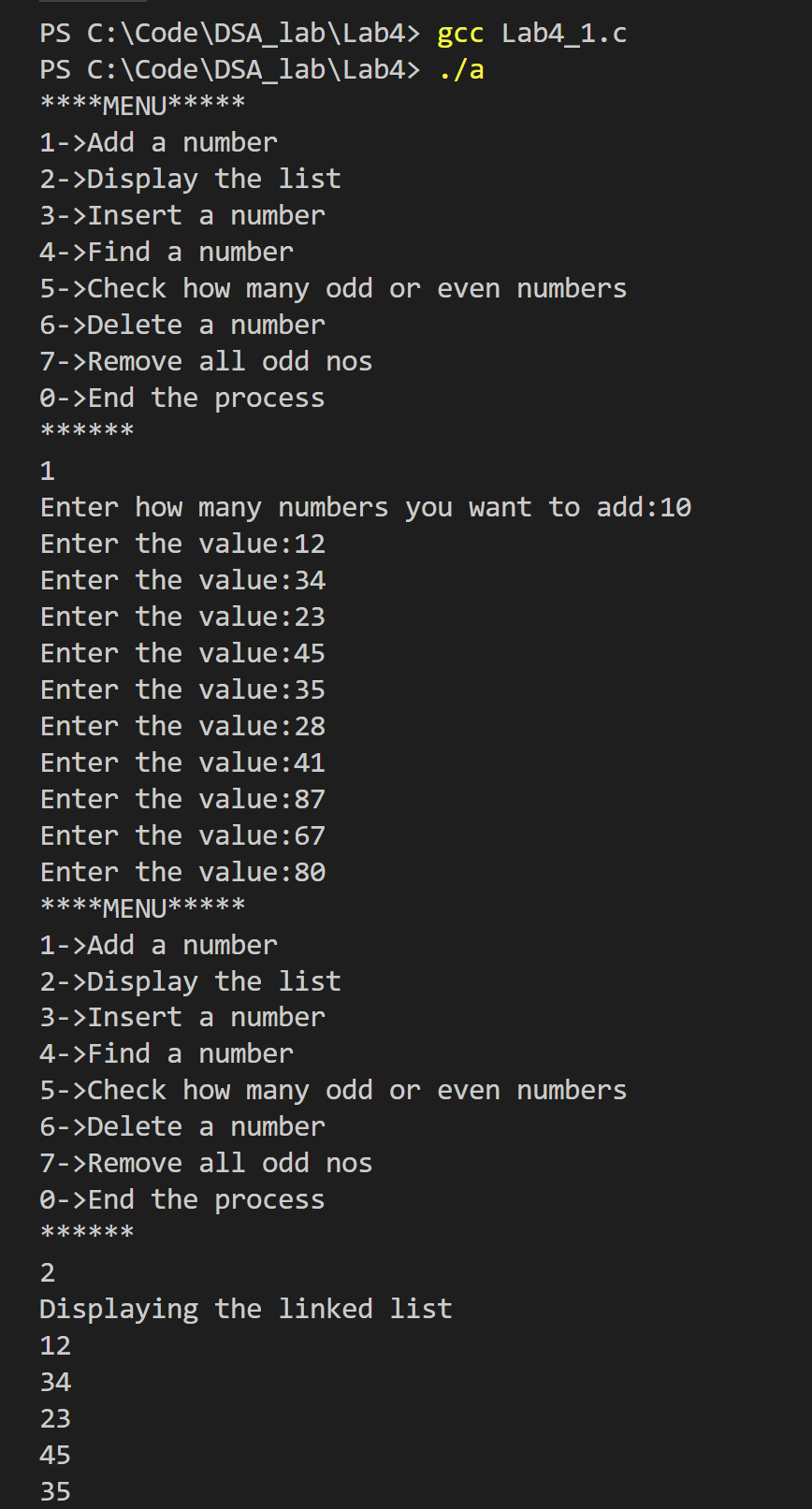
break;

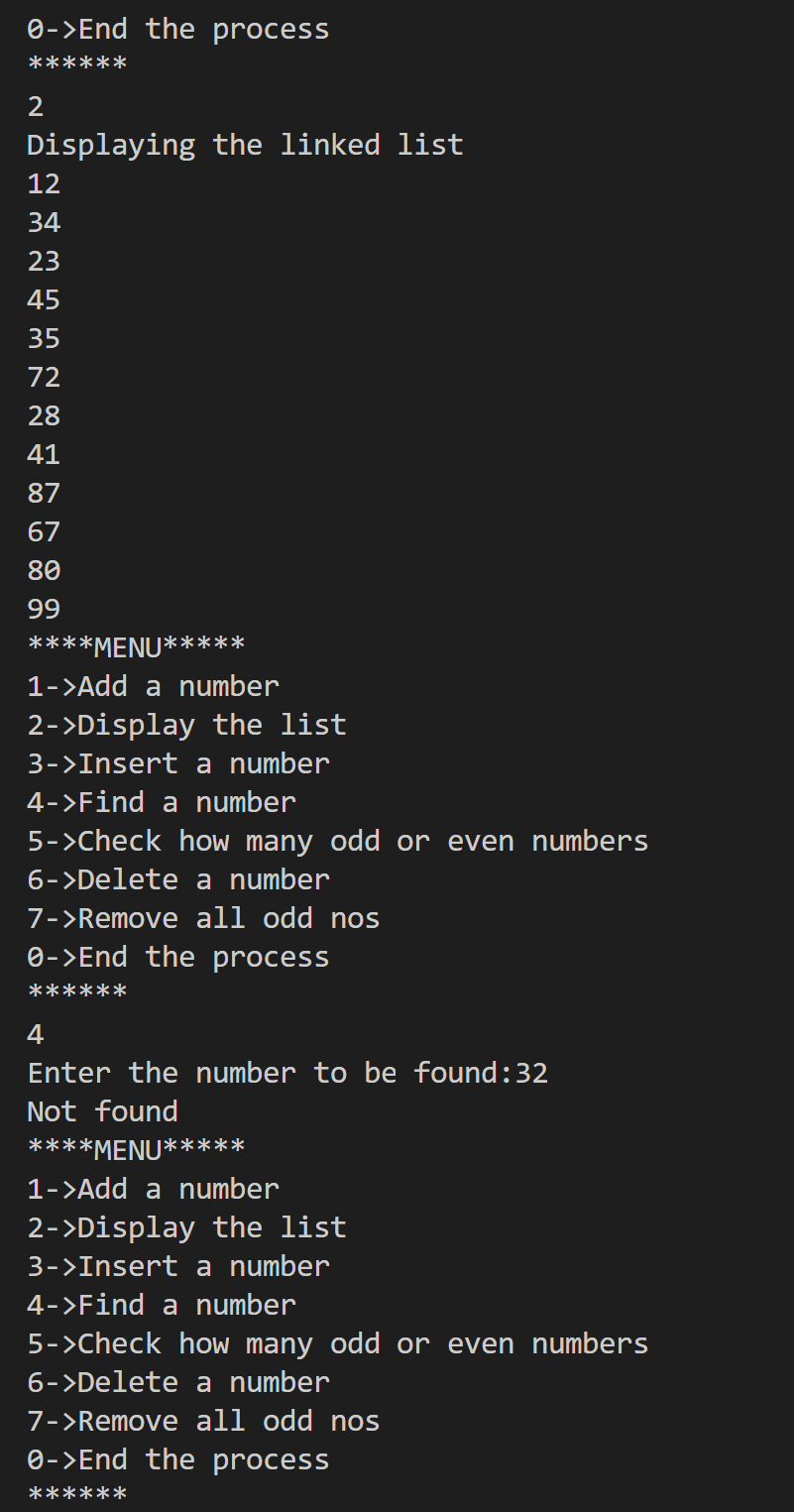
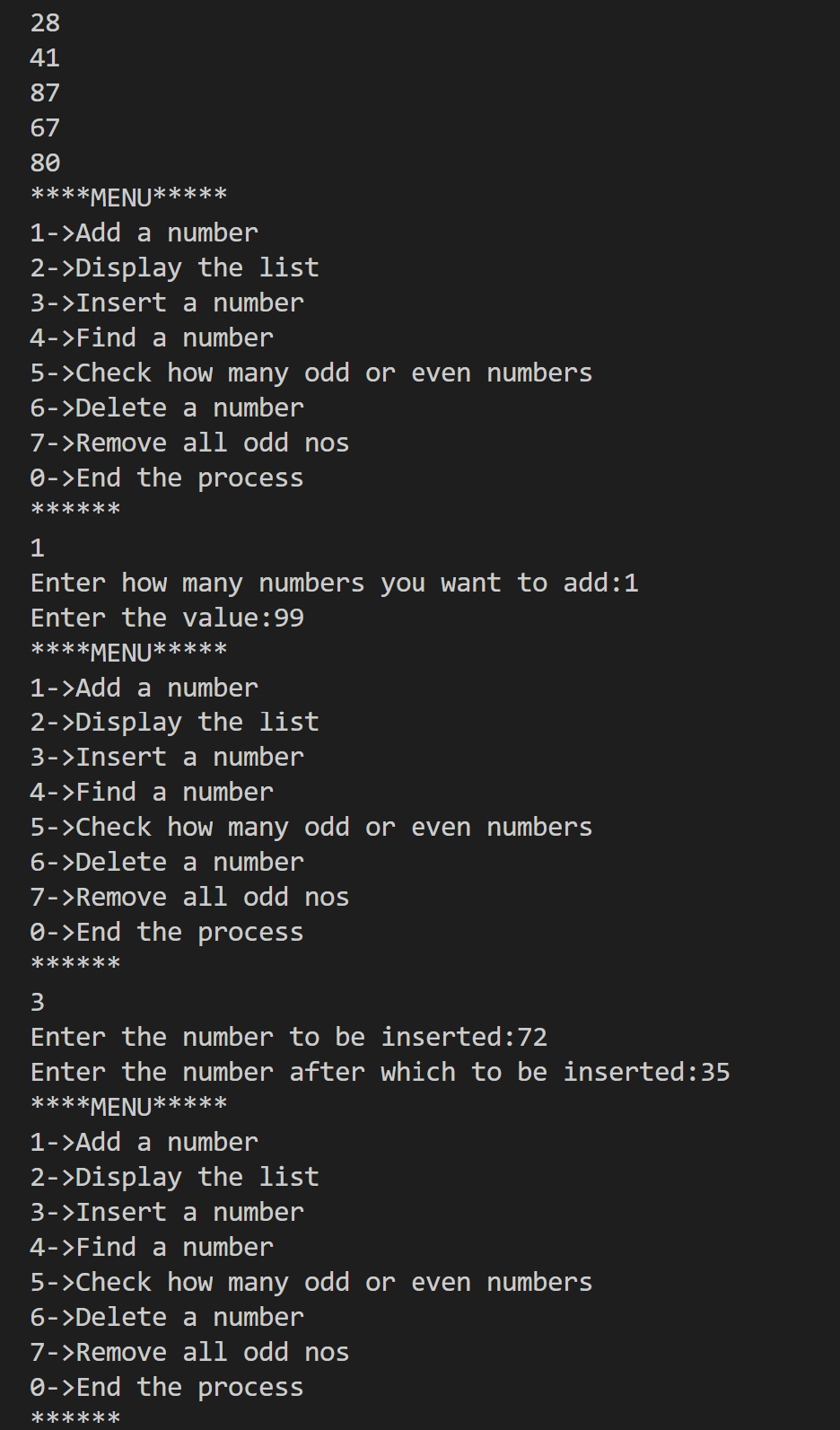
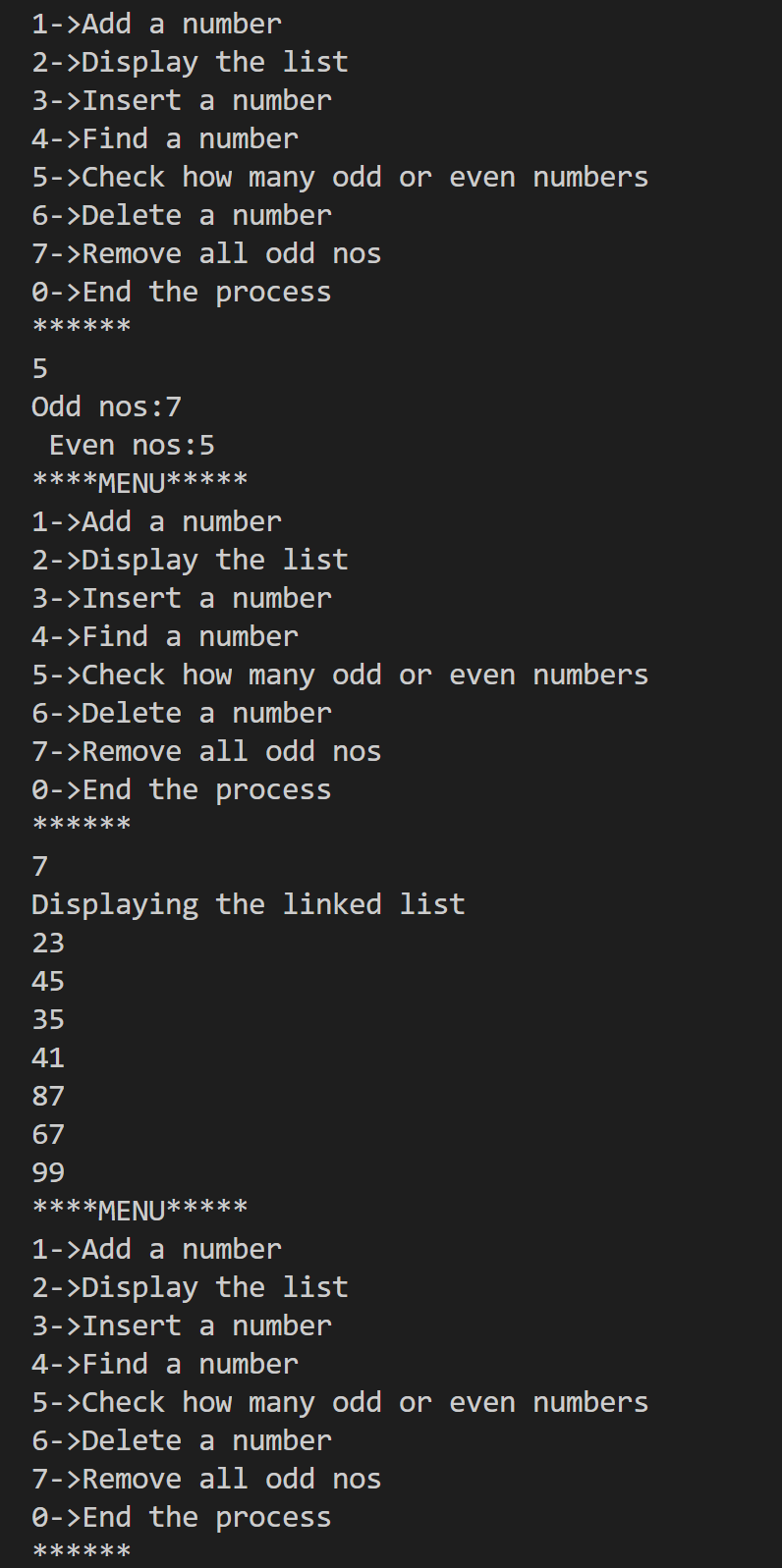
}

}

return 0;

}



**2)**

#include<stdio.h>

#include<stdlib.h>

#define MAX 10

struct Node{

int regno;

char name[50];

float cgpa;

struct Node\* next;

};

void display(struct Node\* head){

struct Node\* temp=(struct Node\*)malloc(sizeof(struct Node));

temp=head->next;

printf("Displaying the linked list\n");

while (temp!=NULL){

printf("%d\n%s\n%f\n",temp->regno,temp->name,temp->cgpa);

temp=temp->next;

}

}

void add(struct Node\* head, struct Node\* n){

struct Node\* temp=(struct Node\*)malloc(sizeof(struct Node));

if (head->next==NULL){

n->next=NULL;

head->next=n;

return;

}

temp=head->next;

while (temp->next!=NULL){

temp=temp->next;

}

temp->next=n;

n->next=NULL;

}

void search(struct Node\* head, int r){

struct Node\* temp=(struct Node\*)malloc(sizeof(struct Node));

temp=head->next;

while(temp!=NULL){

if (temp->regno==r){

printf("%s\n",temp->name);

return;

}

temp=temp->next;

}

printf("Register not found\n");

}

void update(struct Node\* head, int r, float cg){

struct Node\* temp=(struct Node\*)malloc(sizeof(struct Node));

temp=head->next;

while(temp!=NULL){

if(temp->regno==r){

temp->cgpa=cg;

return;

}

temp=temp->next;

}

}

void topper(struct Node\* head){

struct Node\* temp=(struct Node\*)malloc(sizeof(struct Node));

temp=head->next;

struct Node\* topper=(struct Node\*)malloc(sizeof(struct Node));

float t=0;

while (temp!=NULL){

if (temp->cgpa>t){

topper=temp;

t=temp->cgpa;

}

temp=temp->next;

}

printf("%d\n%s\n%f\n",topper->regno,topper->name,topper->cgpa);

}

struct Node\* del(struct Node\* head, int r){

struct Node\* d=(struct Node\*)malloc(sizeof(struct Node));

if (head->next->regno==r){

d=head->next;

head->next=d->next;

return d;

}

struct Node\* temp=(struct Node\*)malloc(sizeof(struct Node));

temp=head->next;

while (temp->next!=NULL){

if (temp->next->regno==r){

d=temp->next;

temp->next=d->next;

return d;

}

temp=temp->next;

}

}

struct Queue{

int front;

int rear;

struct Node\* a[50];

};

int qis\_full(struct Queue\* q){

if (q->front==q->rear && q->rear>-1){

return 1;

}

else{

return 0;

}

}

int qis\_empty(struct Queue\* q){

if (q->rear==-1){

return 1;

}

else{

return 0;

}

}

void enqueue(struct Queue\* q, struct Node\* n){

if (qis\_full(q)){

printf("Queue is full\n");

}

else{

q->rear++;

q->a[q->rear]=n;

}

}

void dequeue(struct Queue\* q){

if (qis\_empty(q)){

printf("Queue is empty");

}

else{

q->front++;

struct Node\* n=(struct Node\*)malloc(sizeof(struct Node));

n=q->a[q->front];

printf("%d\n%s\n%f",n->regno,n->name, n->cgpa);

}

}

void display\_q(struct Queue\* q){

if (qis\_empty(q)){

printf("Queue is empty\n");

}

else{

printf("Displaying the queue:");

for (int i=(q->front+1);i<(q->rear+1);i++){

struct Node\* n=(struct Node\*)malloc(sizeof(struct Node));

n=q->a[i];

printf("%d\n%s\n%f", n->regno, n->name,n->cgpa);

}

}

}

void cgpa\_8(struct Node\* head, struct Queue\* q){

struct Node\* temp=(struct Node\*)malloc(sizeof(struct Node));

temp=head->next;

struct Node\* n=(struct Node\*)malloc(sizeof(struct Node));

while (temp!=NULL){

if (temp->cgpa>8.5){

n=del(head, temp->regno);

enqueue(q,n);

}

temp=temp->next;

}

}

struct Stack{

int top;

struct Node\* a[50];

};

int is\_empty(struct Stack\* s){

if (s->top==-1){

return 1;

}

else{

return 0;

}

}

int is\_full(struct Stack\* s){

if (s->top==(MAX-1)){

return 1;

}

else{

return 0;

}

}

void push(struct Stack\* s, struct Node\* n){

if (is\_full(s)){

printf("Stack is full\n");

}

else{

s->top++;

s->a[s->top]=n;

}

}

void pop(struct Stack\* s){

if (is\_empty(s)){

printf("Stack is empty\n");

}

else{

struct Node\* n=(struct Node\*)malloc(sizeof(struct Node));

n=s->a[s->top];

printf("%d\n%s\n%f",n->regno,n->name,n->cgpa);

s->top--;

}

}

void display\_s(struct Stack\* s){

for (int i=0;i<(s->top+1);i++){

struct Node\* n=(struct Node\*)malloc(sizeof(struct Node));

n=s->a[i];

printf("%d\n%s\n%f", n->regno, n->name, n->cgpa);

}

}

void Nine\_pointer(struct Node\* head, struct Stack\* s){

struct Node\* temp=(struct Node\*)malloc(sizeof(struct Node));

temp=head->next;

struct Node\* n=(struct Node\*)malloc(sizeof(struct Node));

while (temp!=NULL){

if (temp->cgpa>9){

n=del(head, temp->regno);

push(s,n);

}

temp=temp->next;

}

}

int main(){

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

head->next=NULL;

int c=1,n,k=0;

struct Node newnode[30];

int r;

float cg;

struct Queue\* q=(struct Queue\*)malloc(sizeof(struct Queue));

q->rear=-1;

q->front=-1;

struct Stack\* s=(struct Stack\*)malloc(sizeof(struct Stack));

s->top=-1;

/\*newnode[0].regno=1010;

newnode[0].name="harini";

newnode[0].cgpa=9.3;

newnode[1].regno=1009;

newnode[1].name="madhu";

newnode[1].cgpa=8.3;

add(head,&newnode[0]);

add(head,&newnode[1]);

display(head);

cgpa\_8(head, q);

display(head);

display\_q(q);\*/

while(c>0){

printf("\*\*\*\*MENU\*\*\*\*\n");

printf("1->Add student details\n2->Display the list\n3->Search by regno\n");

printf("4->Search for topper\n5->CGPA>8.5\n6->Update CGPA\n");

printf("7->Display queue\n8->Nine pointer stack\n9>Display stack\n");

printf("0->End the process\n");

scanf("%d",&c);

switch(c){

case 1:

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

scanf("%d",&n);

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

//struct Node\* newnode=(struct Node\*)malloc(sizeof(struct Node));

printf("Enter regno, name and cgpa:");

scanf("%d\n%s\n%f",&newnode[k].regno, newnode[k].name, &newnode[k].cgpa);

add(head,&newnode[k]);

k++;

}

break;

case 2:

display(head);

break;

case 3:

printf("Enter the register number:");

scanf("%d",&r);

search(head,r);

break;

case 4:

topper(head);

break;

case 5:

cgpa\_8(head,q);

break;

case 6:

printf("Enter the register number:");

scanf("%d",&r);

printf("Enter the updated cgpa:");

scanf("%f",&cg);

update(head,r,cg);

break;

case 7:

display\_q(q);

break;

case 8:

Nine\_pointer(head,s);

break;

case 9:

display\_s(s);

break;

}

}

}

