Assignment 1.

Submitted by Roll:1906028,

Q 1.Converting a decimal Number into Binary using Double linked list.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Welcome to GDB Online.

GDB online is an online compiler and debugger tool for C, C++, Python, PHP, Ruby,

C#, VB, Perl, Swift, Prolog, Javascript, Pascal, HTML, CSS, JS

Code, Compile, Run and Debug online from anywhere in world.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<stdio.h>

#include<stdlib.h>

typedef struct node sll;

struct node{

int data;

struct node \*next;

struct node \*prev;

};

int Read\_element();

void Insert(sll\*\* ,int);

void traverse(sll\*);

void convert(sll\*\*,int);

void convert(sll \*\*h,int n)

{

int i,s;

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

{

s= n%2;

Insert(&\*h,s);

n=n/2;

}

}

void reverse(sll \*\*h){

sll \*t=NULL;

sll \*c = \*h;

while (c!=NULL)

{

t = c->prev;

c->prev = c->next;

c->next = t;

c = c->prev;

}

if(t != NULL )

\*h = t->prev;

}

void Insert(sll \*\*h,int s){

sll \*temp,\*new;

new=(sll\*)malloc(sizeof(sll));

new->data=s;

new->next=NULL;

new->prev=NULL;

if(\*h==NULL){

\*h=new;

}

else{

temp=\*h;

while(temp->next!=NULL){

temp=temp->next;

}

temp->next=new;

new->prev=temp;

}

}

void traverse(sll \*h){

sll \*t;

if(h==NULL){

printf("List is empty");

}

else{

printf("\nThe binary of the decimal Number is...");

t = h;

while(t!=NULL)

{

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

t = t->next;

}

}

}

int main(){

struct node \*head=NULL;

int n;

printf("Enter the decimal Number\n");

scanf("%d",&n);

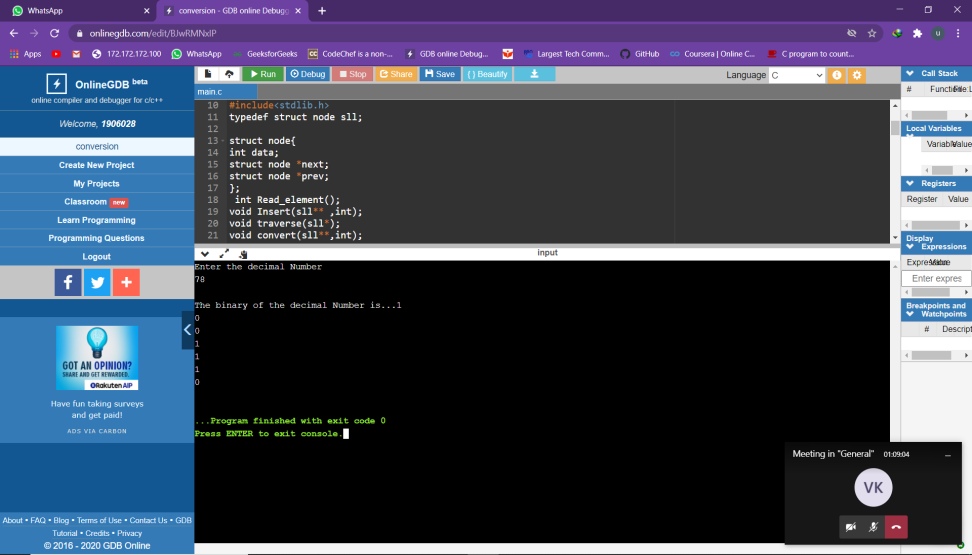
convert(&head,n);

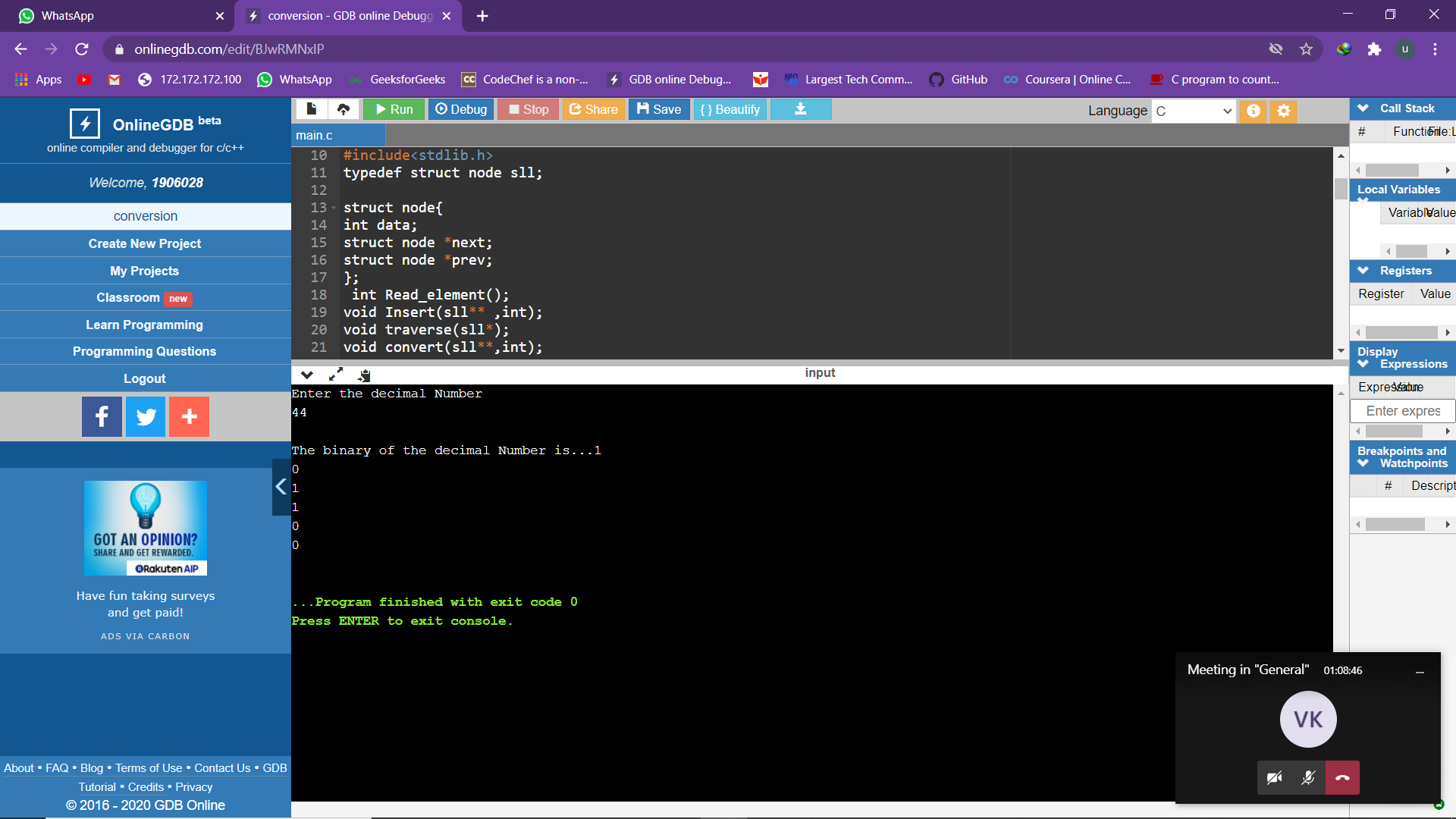
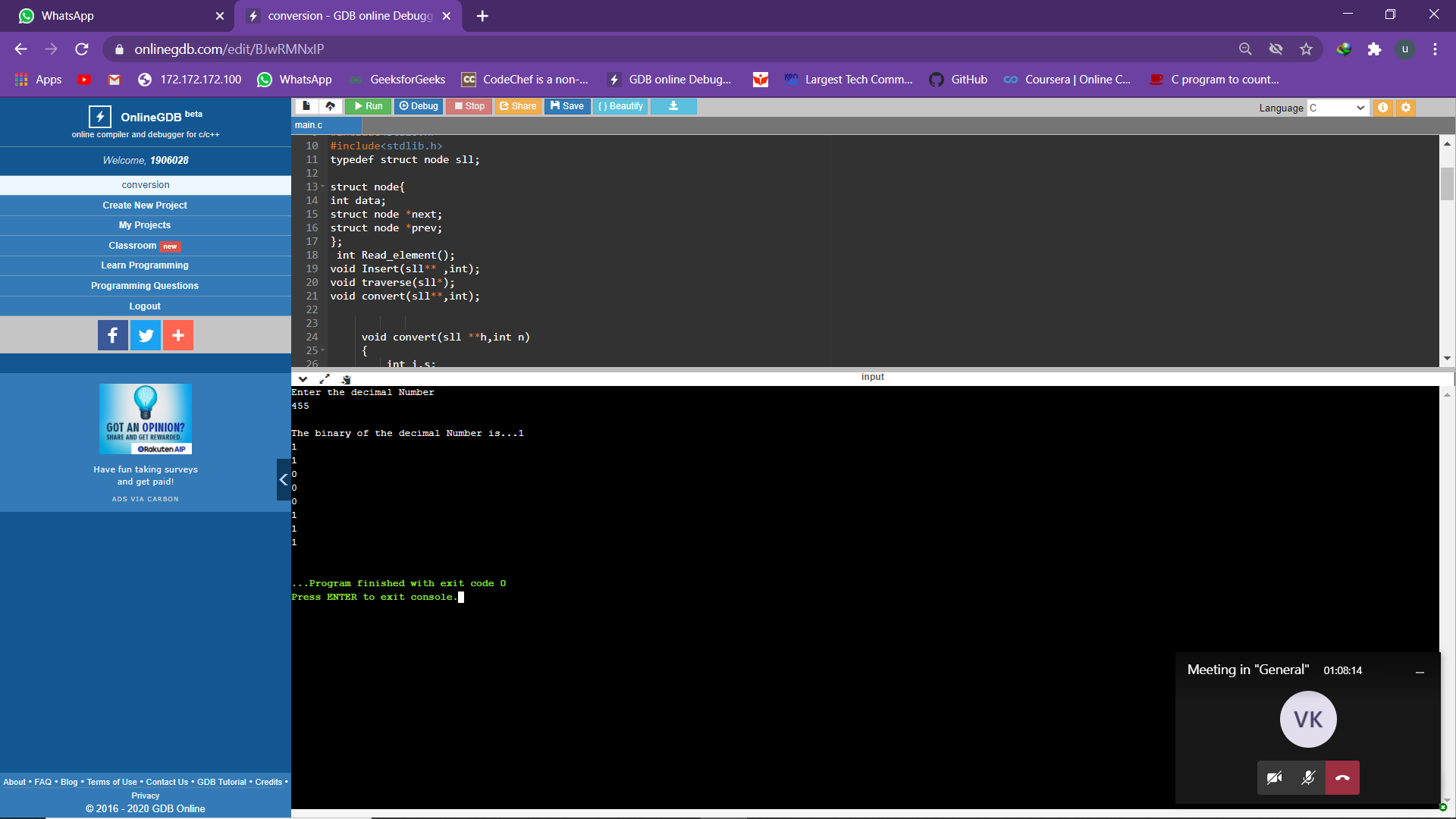
reverse(&head);

traverse(head);

return 0;

}





Qno 2. Taking inputs in ascending order and merging them in single circular list

#include<stdio.h>

#include<stdlib.h>

typedef struct node sll;

struct node{

int data;

struct node \*next;

};

int Read\_element();

void Insert(sll\*\* ,int);

void traverse(sll\*);

void merge(sll\*\*,sll\*\*,sll\*\*);

int Read\_element(){

int a;

scanf("%d",&a);

return a;

}

void Insert(sll \*\*h,int s){

sll \*temp,\*new;

new=(sll\*)malloc(sizeof(sll));

new->data=s;

new->next=new;

if(\*h==NULL){

\*h=new;

new->next=new;

}

else{

temp=\*h;

while(temp->next!=\*h){

temp=temp->next;

}

if(temp->data >new->data) {

printf("Enter in ascending order\n");

}

else{

temp->next=new;

new->next=\*h;

}

}

}

void traverse(sll \*h){

sll \*t;

if(h==NULL){

printf("List is empty");

}

else{

printf("\nThe elements in list are...");

t = h;

while(t->next!=h)

{

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

t = t->next;

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

}

}

void merge(sll \*\*p,sll \*\*q,sll \*\*r){

sll \*t,\*t1,\*t2;

t=\*p;

t1=\*q;

sll \*new,\*temp;

t2=\*r;

while(t->next!=\*p && t1->next!=\*q){

new =(sll\*)malloc(sizeof(sll));

new->next=new;

if(t->data <t1->data){

new->data=t->data;

t=t->next;

}

else{

new->data=t1->data;

t1=t1->next;

}

if(\*r==NULL) {

\*r=new;

temp=new;

}

else{

temp->next=new;

new->next=t2;

temp=new;

}

}

if(t->next!=\*p){

while(t->next!=\*p){

new=(sll\*)malloc(sizeof(sll));

new->next=new;

new->data=t->data;

temp->next=new;

new->next=t2;

temp=new;

t=t->next;

}

}

if(t1->next!=\*q)

{

while(t1->next!=\*q){

new=(sll\*)malloc(sizeof(sll));

new->next=new;

new->data=t1->data;

temp->next=new;

new->next=t2;

temp=new;

t1=t1->next;

}

}

}

int main(){

struct node \*head=NULL;

struct node \*head\_=NULL;

struct node \*m=NULL;

int i,n,p,q,d,s;

printf("Enter 1st single circular list\n ");

printf("Enter no of nodes\n");

scanf("%d",&n);

printf("Enter elements\n");

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

s= Read\_element();

Insert(&head,s);

}

traverse(head);

Insert(&head,9999);

printf("Enter 2nd single circular list\n");

printf("Enter no of nodes\n");

scanf("%d",&n);

printf("Enter elements\n");

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

s= Read\_element();

Insert(&head\_,s);

}

traverse(head\_);

Insert(&head\_,9999);

merge(&head,&head\_,&m);

printf("\n merged list\n");

traverse(m);

return 0;

}

