Anirban Das (001910501077)

BCSE-II

Q1.

CODE:

//printing an integer using pointer and pointer to ponter

#include<stdio.h>

int main(){

int x;

printf("Enter an integer: ");

scanf("%d",&x);

int \*pt1;

pt1=&x;

int \*\*pt2;

pt2=&pt1;

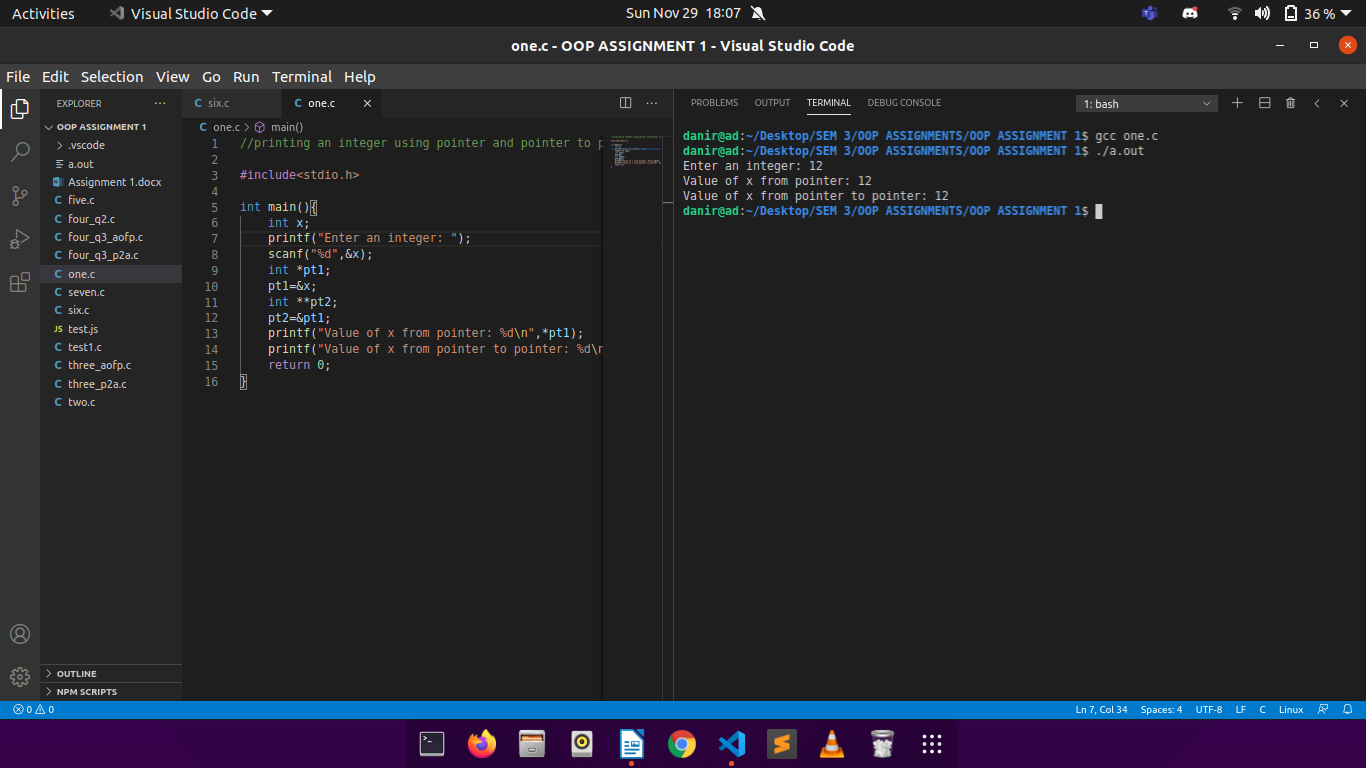
printf("Value of x from pointer: %d\n",\*pt1);

printf("Value of x from pointer to pointer: %d\n",\*(\*pt2));

return 0;

}

OUTPUT:



Q2.

CODE:

//implementing 1-D array using single pointer

#include<stdio.h>

#include<stdlib.h>

int main(){

int n,i;

printf("Enter array size: ");

scanf("%d\n",&n);

int \*a;

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

printf("Enter the values in array:\n");

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

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

}

printf("The 1-D array looks like:\n");

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

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

}

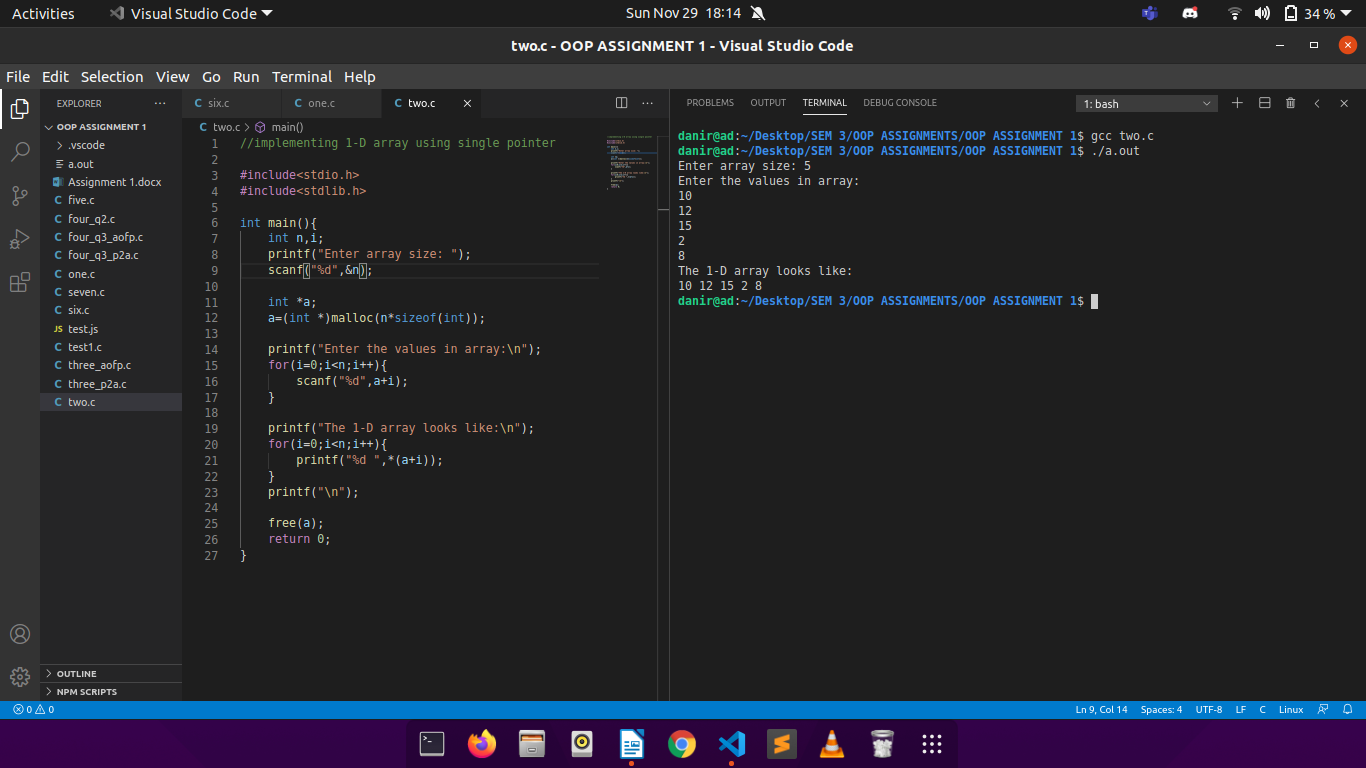
printf("\n");

free(a);

return 0;

}

OUTPUT:



Q3a.

CODE:

//implementing 2-D array using array of pointers

#include<stdio.h>

#include<stdlib.h>

int main(){

int r,c,i,j;

printf("Enter number of rows: ");

scanf("%d",&r);

printf("Enter number of columns: ");

scanf("%d",&c);

int\* a[r];

printf("\nEnter the elements of 2D array:\n");

for(i=0;i<r;i++){

a[i]=(int \*)malloc(c\*sizeof(int));

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

scanf("%d",\*(a+i) + j);

}

printf("\nThe 2D array looks like this:\n");

for(i=0;i<r;i++){

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

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

printf("\n");

}

for(i=0;i<r;i++){

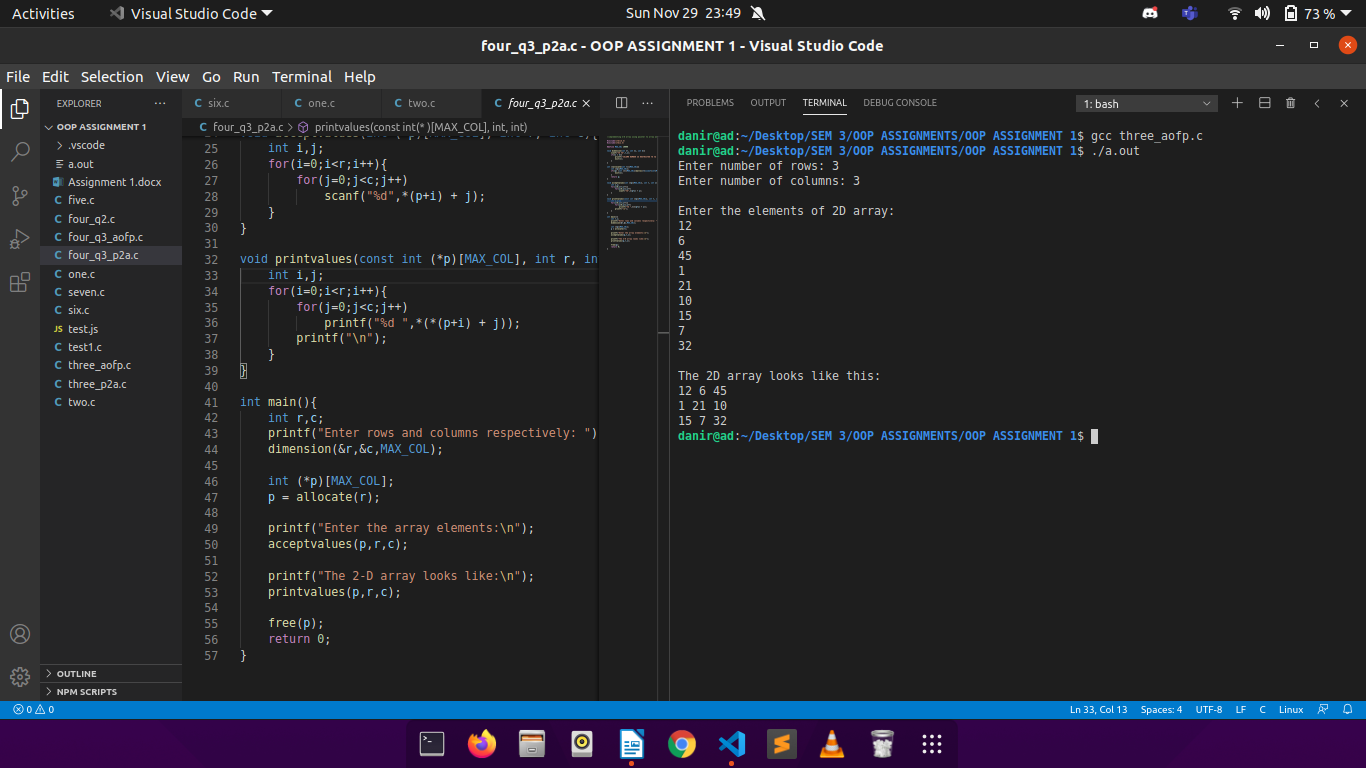
free(a[i]);

}

return 0;

}

OUTPUT:



Q3b.

CODE:

//implementing 2-D array using pointer to array

#include<stdio.h>

#include<stdlib.h>

int main(){

int r,c,i,j;

printf("Enter number of rows: ");

scanf("%d",&r);

printf("Enter number of columns: ");

scanf("%d",&c);

int (\*p)[c];

p=(int (\*)[])malloc(r\*sizeof(int [c]));

printf("\nEnter the elements of 2D array:\n");

for(i=0;i<r;i++){

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

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

}

printf("\nThe 2D array looks like this:\n");

for(i=0;i<r;i++){

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

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

printf("\n");

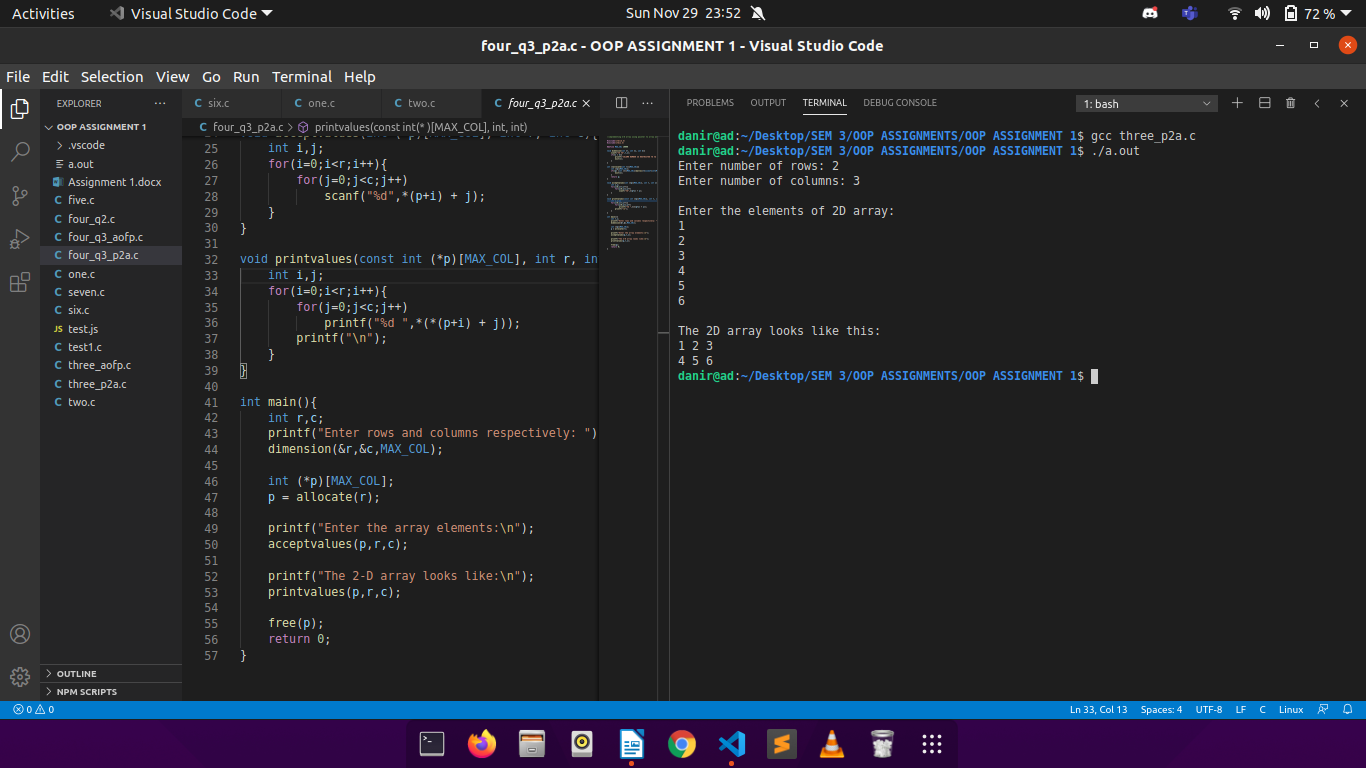
}

free(p);

return 0;

}

OUTPUT:



Q4a.

CODE:

// implementing 1-D array using single pointer and functions

#include<stdio.h>

#include<stdlib.h>

void dimension(int \*n){

scanf("%d",n);

}

void allocate(int \*\*p, int n){

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

}

void acceptvalues(int \*a, int n){

int i;

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

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

}

}

void printvalues(int \*a, int n){

int i;

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

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

}

printf("\n");

}

void FREE(int \*p){

free(p);

}

int main(){

int n;

printf("Enter array size: ");

dimension(&n);

int \*p;

allocate(&p,n);

printf("Enter the values in array:\n");

acceptvalues(p,n);

printf("The 1-D array looks like:\n");

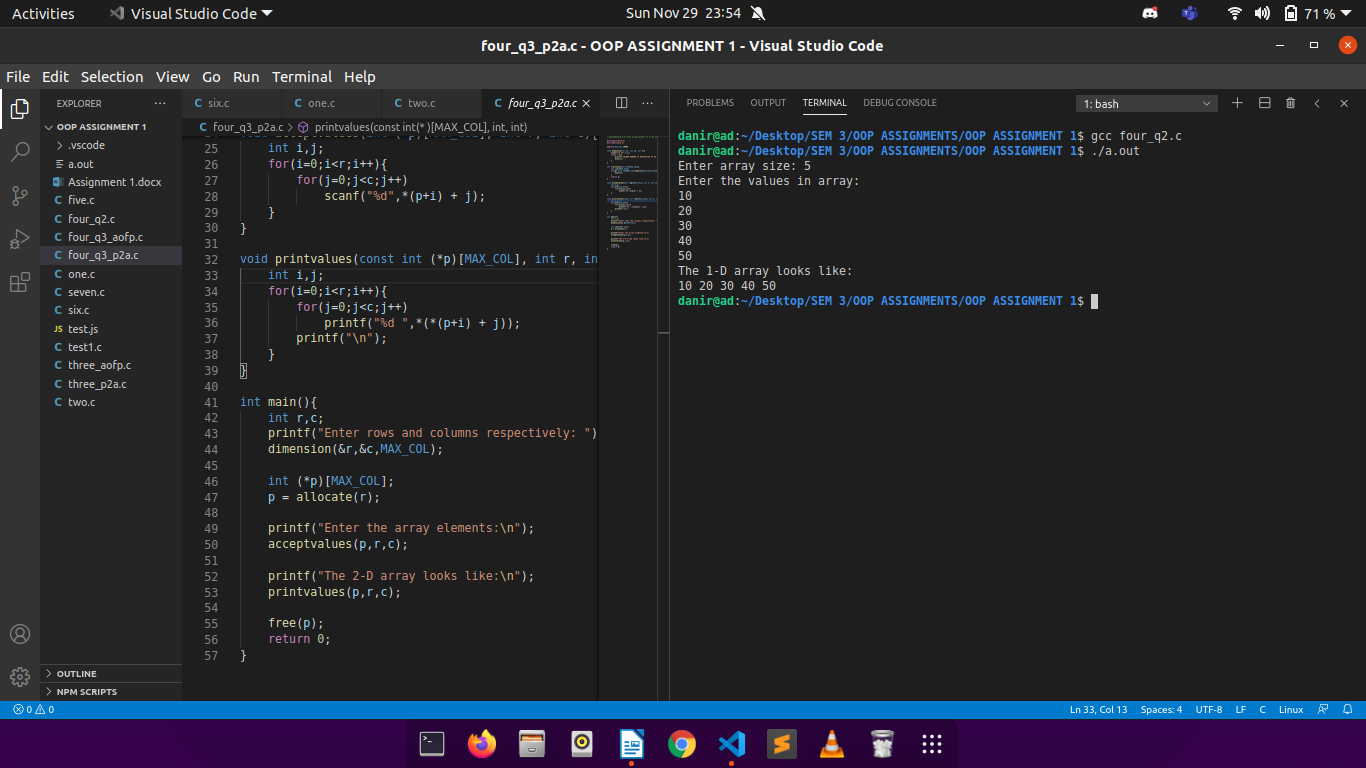
printvalues(p,n);

FREE(p);

return 0;

}

OUTPUT:



Q4b.

CODE:

//implementing 2-D array using array of pointers and functions

#include<stdlib.h>

#include<stdio.h>

void dimension(int \*r, int \*c){

scanf("%d %d",r,c);

}

void allocate(int \*\*\*p){

\*p=(int \*\*)malloc(100\*sizeof(int \*));

}

void acceptvalues(int \*\*p, int r, int c){

int i,j;

for(i=0;i<r;i++){

p[i]=(int \*)malloc(c\*sizeof(int));

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

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

}

}

void printvalues(int \*\*p,int r,int c){

int i,j;

for(i=0;i<r;i++){

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

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

printf("\n");

}

}

void FREE(int \*\*p, int r){

int i;

for(i=0;i<r;i++){

free(p[i]);

}

free(p);

}

int main(){

int r,c;

printf("Enter row and column number respectivey: ");

dimension(&r,&c);

int \*\*p;

allocate(&p);

printf("\nEnter the elements of 2D array:\n");

acceptvalues(p,r,c);

printf("\nThe 2D array looks like this:\n");

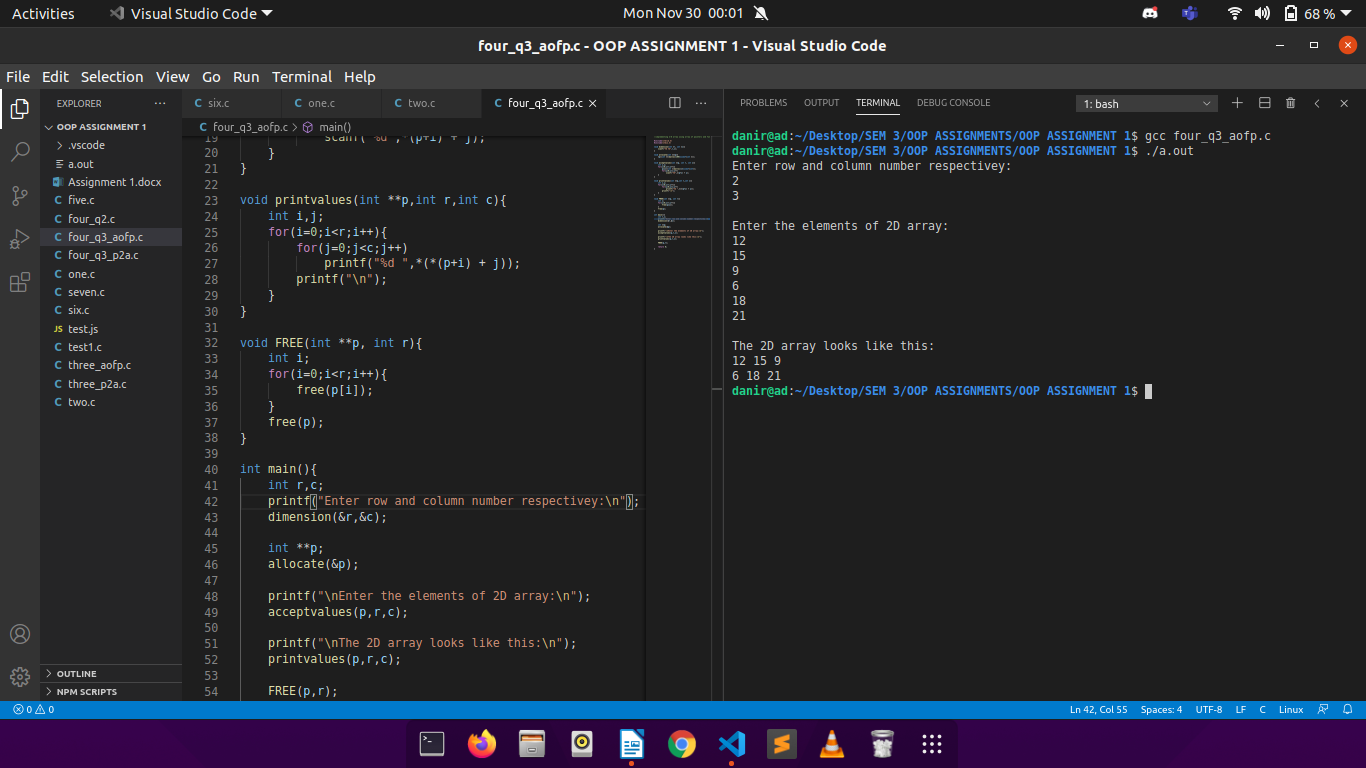
printvalues(p,r,c);

FREE(p,r);

return 0;

}

OUTPUT:



Q4c.

CODE:

//implementing 2-D array using pointer to array and functions

#include<stdlib.h>

#include<stdio.h>

#define MAX\_COL 10000

void dimension(int \*r, int \*c, int C){

scanf("%d %d",r,c);

if(\*c > C){

printf("COLUMN NUMBER IS RESTRICTED TO %d !!!\n",C);

exit(1);

}

}

int (\*allocate(int r))[MAX\_COL]{

int (\*p)[MAX\_COL];

if((p= (int (\*)[MAX\_COL])malloc(r\*sizeof(int[MAX\_COL]))) == NULL ){

exit(1);

}

return p;

}

void acceptvalues(int (\*p)[MAX\_COL], int r, int c){

int i,j;

for(i=0;i<r;i++){

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

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

}

}

void printvalues(const int (\*p)[MAX\_COL], int r, int c){

int i,j;

for(i=0;i<r;i++){

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

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

printf("\n");

}

}

int main(){

int r,c;

printf("Enter rows and columns respectively: ");

dimension(&r,&c,MAX\_COL);

int (\*p)[MAX\_COL];

p = allocate(r);

printf("Enter the array elements:\n");

acceptvalues(p,r,c);

printf("The 2-D array looks like:\n");

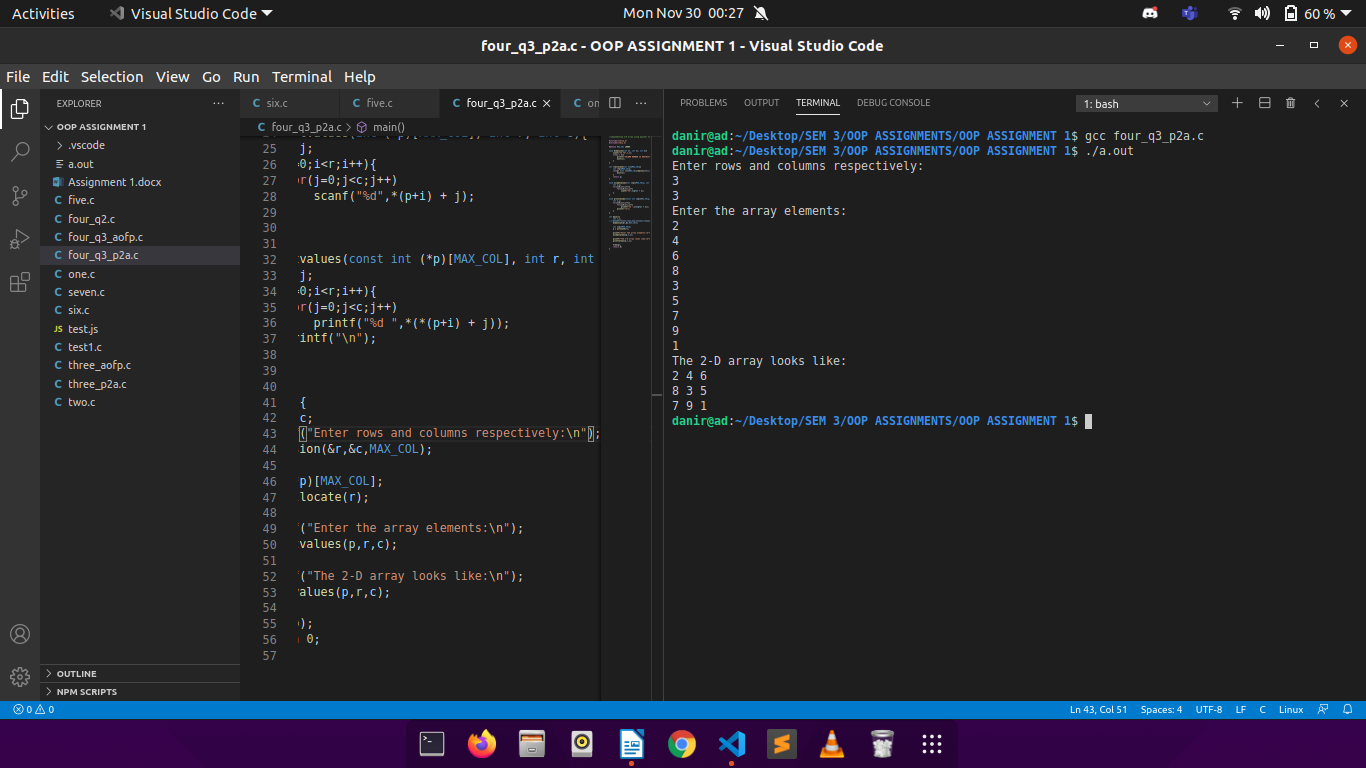
printvalues(p,r,c);

free(p);

return 0;

}

OUTPUT:



Q5.

CODE:

//implementing list of students without a structure

#include<stdio.h>

#include<string.h>

void readData(char name[][31],int age[],int n){

printf("Enter Names & Ages:\n");

int i;

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

scanf("%s",name[i]);

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

}

}

void sortAge(char name[][31],int age[], int n){

char ar[31];

int i,j,temp;

for(i=0;i<n-1;i++){

for(j=0;j<n-i-1;j++){

if(age[j]>age[j+1]){

temp=age[j];

age[j]=age[j+1];

age[j+1]=temp;

strcpy(ar,name[j]);

strcpy(name[j],name[j+1]);

strcpy(name[j+1],ar);

}

}

}

}

void printData(char name[][31],int age[], int n){

printf("The data in ascending order of age is as follows:\n");

int i;

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

printf("%s %d\n",name[i],age[i]);

}

}

int main(){

char name[100][31];

int age[100];

int n;

printf("Enter number of persons: ");

scanf("%d",&n);

readData(name,age,n);

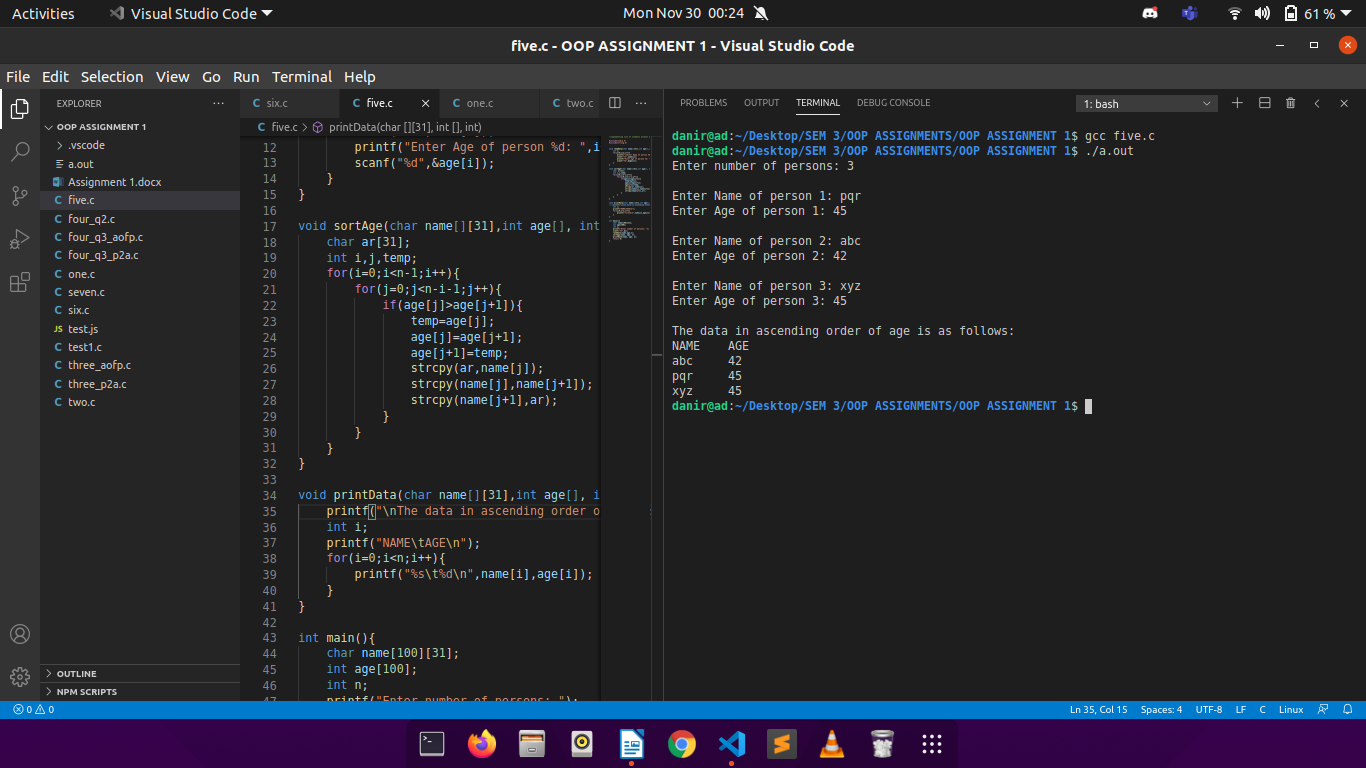
sortAge(name, age, n);

printData(name, age, n);

return 0;

}

OUTPUT:



Q6.

CODE:

//implementing list of students using a structure

#include<stdio.h>

struct student {

int age;

char name[100];

};

void readData(struct student l[], int n){

int i;

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

printf("Enter Name of Student %d: ",i+1);

scanf("%s",l[i].name);

printf("Enter Age of Student %d: ",i+1);

scanf("%d",&l[i].age);

printf("\n");

}

}

void sortAge(struct student l[], int n){

struct student t;

int i,j;

for(i=0;i<n-1;i++){

for(j=0;j<n-i-1;j++){

if(l[j].age>l[j+1].age){

t=l[j];

l[j]=l[j+1];

l[j+1]=t;

}

}

}

}

void printData(struct student l[], int n){

printf("The data in ascending order of age is as follows:\n");

int i;

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

printf("%s %d\n",l[i].name,l[i].age);

}

}

int main(){

int n,i;

printf("Enter number of students: ");

scanf("%d",&n);

struct student ar[n];

readData(ar, n);

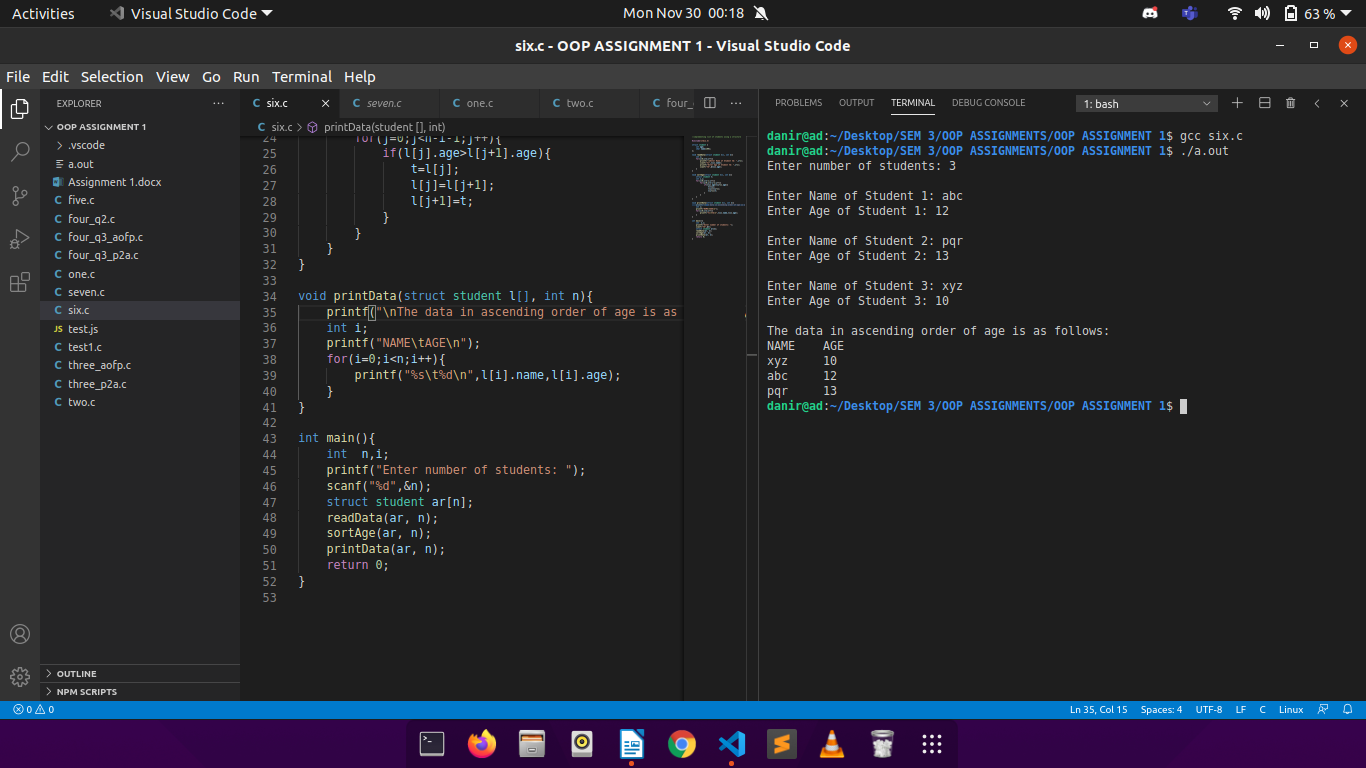
sortAge(ar, n);

printData(ar, n);

return 0;

}

OUTPUT:



Q7.

CODE:

#include<stdio.h>

#include<stdlib.h>

typedef struct {

int roll;

char name[31];

int marks;

} student;

struct node{

student s;

struct node \*adr;

};

struct node \*root;

struct node\* createnode(struct node \*\*);

void addstudent(struct node \*, int);

void display();

struct node \*createnode(struct node \*\*temp){

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

return \*temp;

}

void addstudent(struct node \*temp, int i){

printf("Enter the roll of student %d:\n",i);

scanf("%d",&temp->s.roll);

printf("Enter the name of student %d:\n",i);

scanf("%s",temp->s.name);

printf("Enter the marks obtained by student %d:\n",i);

scanf("%d",&temp->s.marks);

}

void del() {

struct node \*ptr=NULL;

struct node \*prev=NULL;

struct node \*current=NULL;

int roll\_no;

printf("Enter Roll Number to Delete:\n");

scanf("%d",&roll\_no);

if(root->s.roll == roll\_no){

struct node \*tmp;

if(root == NULL)

printf("LIST EMPTY ALREADY!!\n");

tmp = root;

root = (root)->adr;

free(tmp);

}

else{

prev=root;

current=root;

while(current->s.roll!=roll\_no){

prev=current;

current=current->adr;

}

prev->adr = current->adr;

current->adr=NULL;

free(current);

}

printf("Recored Deleted\n\n");

}

void display() {

struct node \*temp;

temp=root;

printf("ROLL NAME MARKS\n");

while(temp) {

printf("-> %d %s %d " , temp->s.roll, temp->s.name, temp->s.marks);

temp=temp->adr;

}

printf("\n");

}

int main(){

int n,i;

struct node \*p;

printf("ENTER NUMBER OF STUDENTS: ");

scanf("%d",&n);

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

struct node \*temp;

temp = createnode(&temp);

addstudent(temp,i+1);

if(root==NULL){

root=temp;

p=temp;

}

else{

p->adr=temp;

p=temp;

}

}

p->adr=NULL;

printf("\nTHE LINKED LIST LOOKS LIKE THIS:\n");

display();

printf("\nWANT TO DELETE A RECORD? PRESS 1.\n");

int ch;

scanf("%d",&ch);

while(ch==1){

if(root==NULL){

printf("LIST EMPTY ALREADY!!\n");

break;

}

del();

display();

printf("\nWANT TO DELETE MORE? PRESS 1 TO CONTINUE AND 0 TO STOP!\n");

scanf("%d",&ch);

}

printf("\nFINAL LIST LOOKS LIKE THIS:\n");

display();

return 0;

}

OUTPUT:

