**STUDENT NAME: P. HASINI**

**REG NUMBER : 192111535**

**SUBJECT NAME : OPERATING SYSTEM WITH DESIGN PRINCIPLES**

**SUBJECT CODE : CSA0470**

DAY 3 PROGRAMS(SEP 28,2022)

**1.SINGLE LEVEL DIRECTORY**

#include<conio.h>

#include<string.h>

void main()

{

int nf=0,i=0,j=0,ch;

char mdname[10],fname[10][10],name[10];

printf("Enter the directory name:");

scanf("%s",mdname);

printf("Enter the number of files:");

scanf("%d",&nf);

do

{

printf("Enter file name to be created:");

scanf("%s",name);

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

{

if(!strcmp(name,fname[i]))

break;

}

if(i==nf)

{

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

nf++;

}

else

printf("There is already %s\n",name);

printf("Do you want to enter another file(yes - 1 or no - 0):");

scanf("%d",&ch);

}

while(ch==1);

printf("Directory name is:%s\n",mdname);

printf("Files names are:");

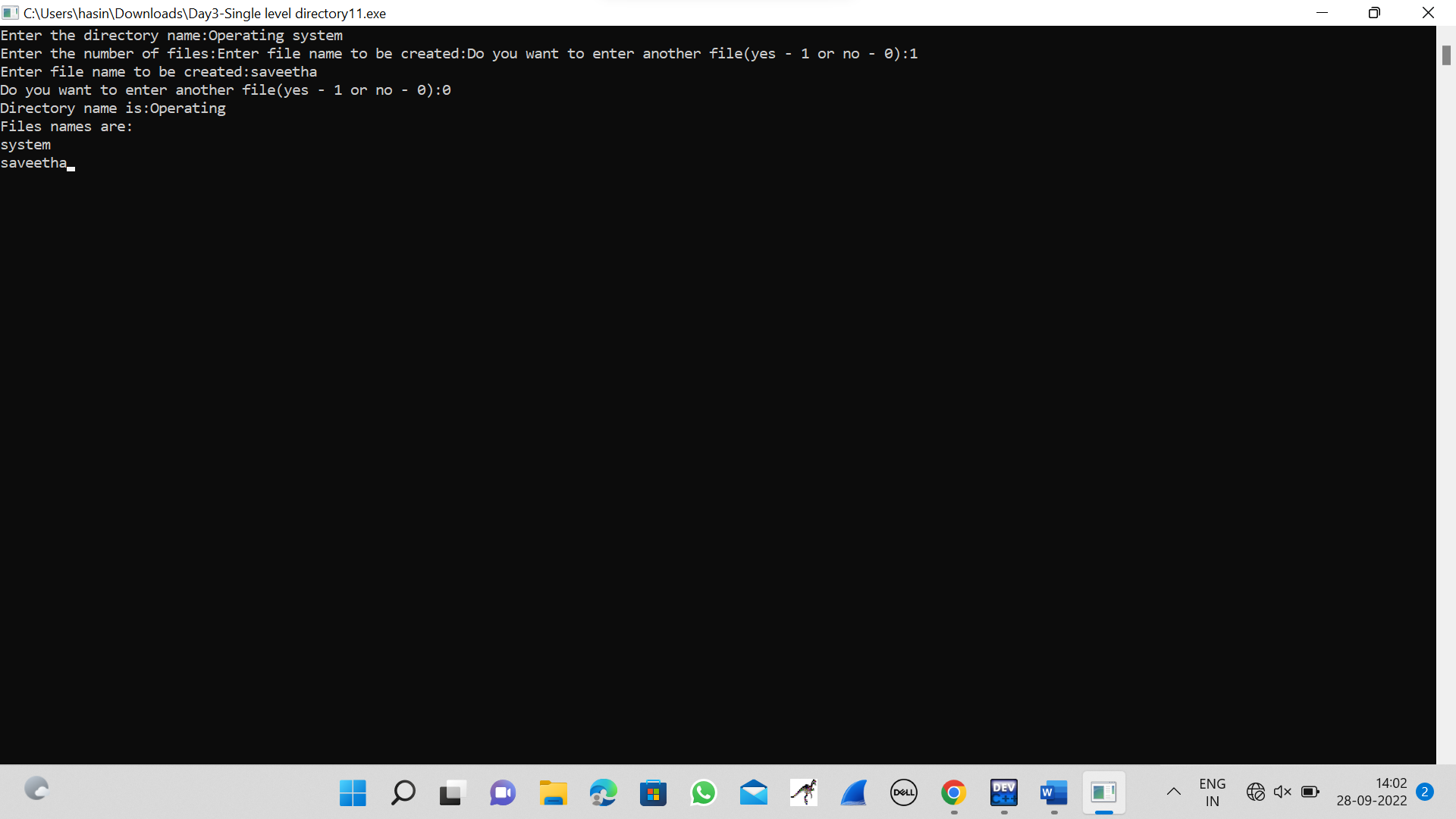
for(i=0;i<j;i++)

printf("\n%s",fname[i]);

getch();

}

**OUTPUT**

****

**2.TWO LEVEL DIRECTORY STRUCTURE**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

struct

{

char dname[10],fname[10][10];

int fcnt;

}dir[10];

void main()

{

int i,ch,dcnt,k;

char f[30], d[30];

dcnt=0;

while(1)

{

printf("1. Create Directory\t2. Create File\t3. Delete File");

printf("\n4. Search File\t\t5. Display\t6. Exit\tEnter your choice --");

scanf("%d",&ch);

switch(ch)

{

case 1: printf("\nEnter name of directory -- ");

scanf("%s", dir[dcnt].dname);

dir[dcnt].fcnt=0;

dcnt++;

printf("Directory created");

break;

case 2: printf("\nEnter name of the directory -- ");

scanf("%s",d);

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

if(strcmp(d,dir[i].dname)==0)

{

printf("Enter name of the file -- ");

scanf("%s",dir[i].fname[dir[i].fcnt]);

dir[i].fcnt++;

printf("File created");

break;

}

if(i==dcnt)

printf("Directory %s not found",d);

break;

case 3: printf("\nEnter name of the directory -- ");

scanf("%s",d);

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

{

if(strcmp(d,dir[i].dname)==0)

{

printf("Enter name of the file -- ");

scanf("%s",f);

for(k=0;k<dir[i].fcnt;k++)

{

if(strcmp(f, dir[i].fname[k])==0)

{

printf("File %s is deleted ",f);

dir[i].fcnt--;

strcpy(dir[i].fname[k],dir[i].fname[dir[i].fcnt]);

goto jmp;

}

}

printf("File %s not found",f);

goto jmp;

}

}

printf("Directory %s not found",d);

jmp : break;

case 4: printf("\nEnter name of the directory -- ");

scanf("%s",d);

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

{

if(strcmp(d,dir[i].dname)==0)

{

printf("Enter the name of the file -- ");

scanf("%s",f);

for(k=0;k<dir[i].fcnt;k++)

{

if(strcmp(f, dir[i].fname[k])==0)

{

printf("File %s is found ",f);

goto jmp1;

}

}

printf("File %s not found",f);

goto jmp1;

}

}

printf("Directory %s not found",d);

jmp1: break;

case 5: if(dcnt==0)

printf("\nNo Directory's ");

else

{

printf("\nDirectory\tFiles");

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

{

printf("\n%s\t\t",dir[i].dname);

for(k=0;k<dir[i].fcnt;k++)

printf("\t%s",dir[i].fname[k]);

}

}

break;

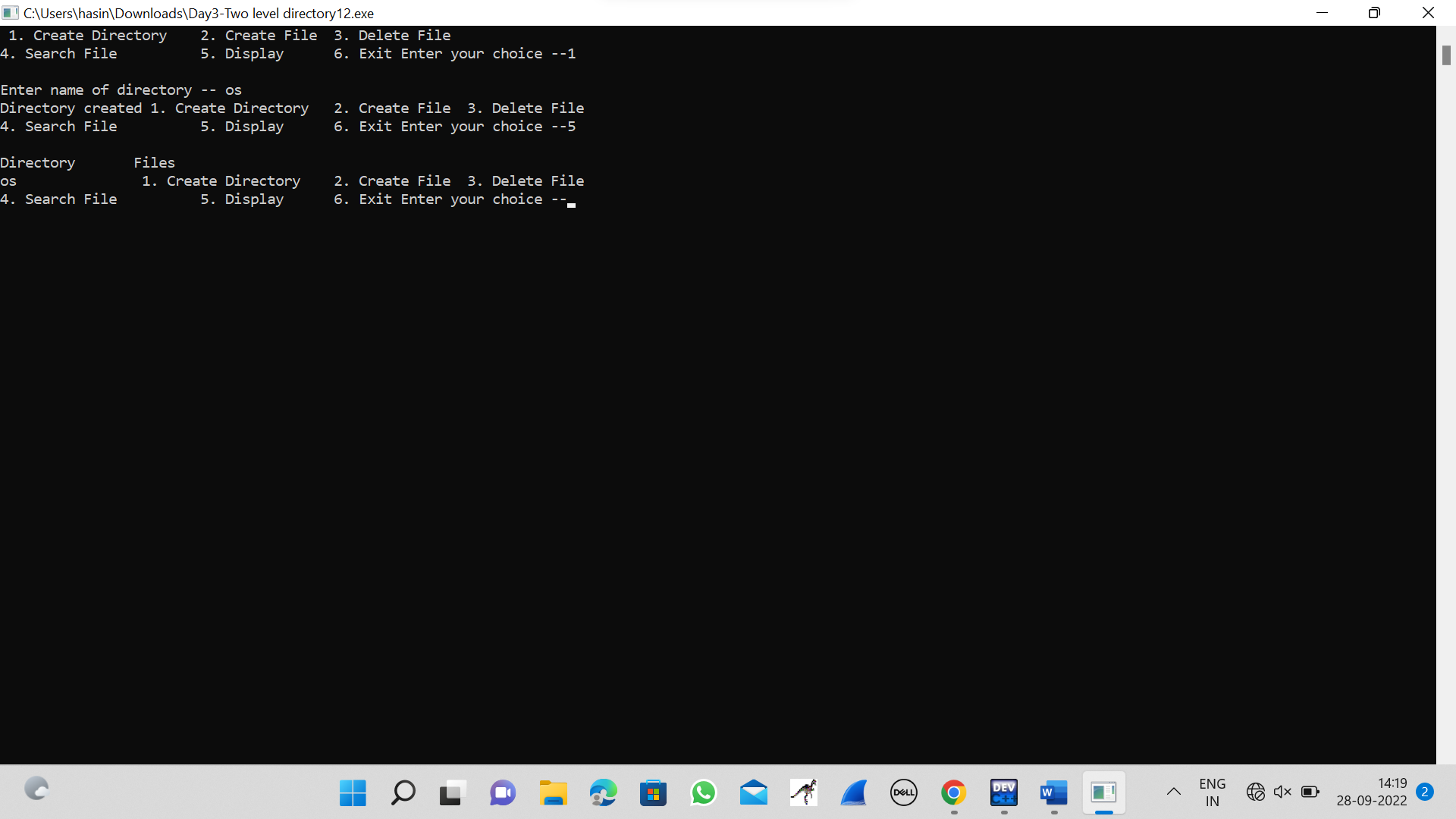
default:exit(0);

}

}

}

**OUTPUT**

****

**13.RANDAM ACCESS FILE**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <windows.h>

struct emp {

char name[50];

float salary;

int age;

int id;

};

struct emp e;

long int size = sizeof(e);

COORD cord = { 0, 0 };

void gotoxy(int x, int y)

{

cord.X = x;

cord.Y = y;

SetConsoleCursorPosition(

GetStdHandle(STD\_OUTPUT\_HANDLE),

cord);

}

FILE \*fp, \*ft;

void addrecord()

{

system("cls");

fseek(fp, 0, SEEK\_END);

char another = 'y';

while (another == 'y') {

printf("\nEnter Name : ");

scanf("%s", e.name);

printf("\nEnter Age : ");

scanf("%d", &e.age);

printf("\nEnter Salary : ");

scanf("%f", &e.salary);

printf("\nEnter EMP-ID : ");

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

fwrite(&e, size, 1, fp);

printf("\nWant to add another"

" record (Y/N) : ");

fflush(stdin);

scanf("%c", &another);

}

}

void deleterecord()

{

system("cls");

char empname[50];

char another = 'y';

while (another == 'y') {

printf("\nEnter employee "

"name to delete : ");

scanf("%s", empname);

ft = fopen("temp.txt", "wb");

rewind(fp);

while (fread(&e, size,

1, fp)

== 1) {

if (strcmp(e.name,

empname)

!= 0)

fwrite(&e, size, 1, ft);

}

fclose(fp);

fclose(ft);

remove("data.txt");

rename("temp.txt", "data.txt");

fp = fopen("data.txt", "rb+");

printf("\nWant to delete another"

" record (Y/N) :");

fflush(stdin);

another = getche();

}

}

void displayrecord()

{

system("cls");

rewind(fp);

printf("\n========================="

"==========================="

"======");

printf("\nNAME\t\tAGE\t\tSALARY\t\t"

"\tID\n",

e.name, e.age,

e.salary, e.id);

printf("==========================="

"==========================="

"====\n");

while (fread(&e, size, 1, fp) == 1)

printf("\n%s\t\t%d\t\t%.2f\t%10d",

e.name, e.age, e.salary, e.id);

printf("\n\n\n\t");

system("pause");

}

void modifyrecord()

{

system("cls");

char empname[50];

char another = 'y';

while (another == 'y') {

printf("\nEnter employee name"

" to modify : ");

scanf("%s", empname);

rewind(fp);

while (fread(&e, size, 1, fp) == 1) {

if (strcmp(e.name, empname) == 0) {

printf("\nEnter new name:");

scanf("%s", e.name);

printf("\nEnter new age :");

scanf("%d", &e.age);

printf("\nEnter new salary :");

scanf("%f", &e.salary);

printf("\nEnter new EMP-ID :");

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

fseek(fp, -size, SEEK\_CUR);

fwrite(&e, size, 1, fp);

break;

}

}

printf("\nWant to modify another"

" record (Y/N) :");

fflush(stdin);

scanf("%c", &another);

}

}

int main()

{

int choice;

fp = fopen("data.txt", "rb+");

if (fp == NULL) {

fp = fopen("data.txt", "wb+");

if (fp == NULL) {

printf("\nCannot open file...");

exit(1);

}

}

system("Color 3F");

printf("\n\n\n\n\t\t\t\t============="

"============================="

"===========");

printf("\n\t\t\t\t~~~~~~~~~~~~~~~~~~~"

"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"

"~~~~~");

printf("\n\t\t\t\t==================="

"============================="

"=====");

printf("\n\t\t\t\t[|:::>:::>:::>::> "

"EMPLOYEE RECORD <::<:::<:::"

"<:::|]\t");

printf("\n\t\t\t\t==================="

"============================="

"=====");

printf("\n\t\t\t\t~~~~~~~~~~~~~~~~~~~~"

"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"

"~~~");

printf("\n\t\t\t\t====================="

"==============================\n");

printf("\n\n\n\t\t\t\t\t\t\t\t\t\t"

"Developer : @Sushant\_Gaurav"

"\n\n\t\t\t\t");

system("pause");

while (1) {

system("cls");

gotoxy(30, 10);

printf("\n1. ADD RECORD\n");

gotoxy(30, 12);

printf("\n2. DELETE RECORD\n");

gotoxy(30, 14);

printf("\n3. DISPLAY RECORDS\n");

gotoxy(30, 16);

printf("\n4. MODIFY RECORD\n");

gotoxy(30, 18);

printf("\n5. EXIT\n");

gotoxy(30, 20);

printf("\nENTER YOUR CHOICE...\n");

fflush(stdin);

scanf("%d", &choice);

switch (choice) {

case 1:

addrecord();

break;

case 2:

deleterecord();

break;

case 3:

displayrecord();

break;

case 4:

modifyrecord();

break;

case 5:

fclose(fp);

exit(0);

break;

default:

printf("\nINVALID CHOICE...\n");

}

}

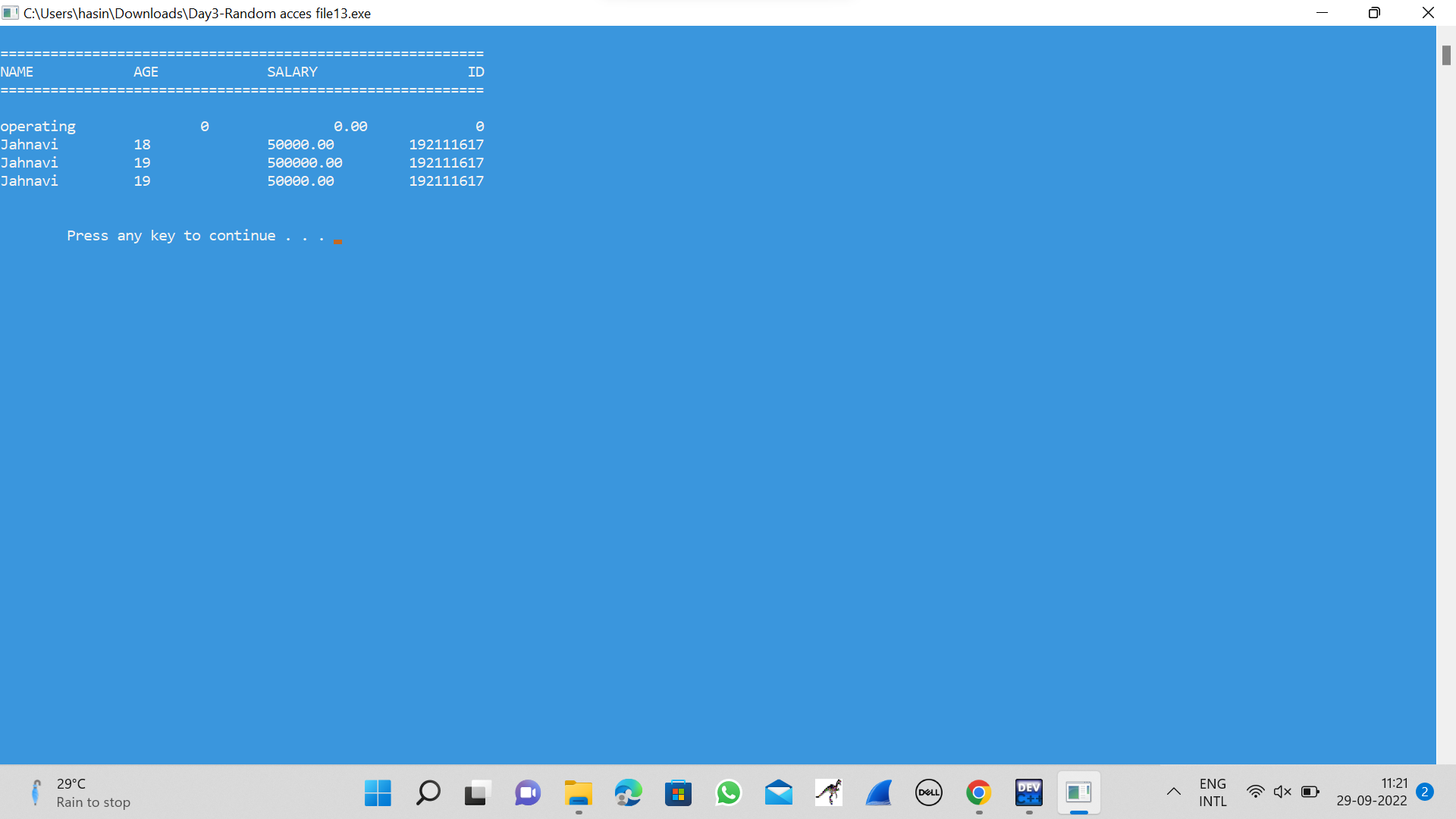
{

return 0;

}

}

**OUTPUT**

****

**4.BANKERS ALGORITHM**

#include <stdio.h>

#include <conio.h>

int main()

{

int Max[10][10], need[10][10], alloc[10][10], avail[10], completed[10], safeSequence[10];

int p, r, i, j, process, count;

count = 0;

printf("Enter the no of processes : ");

scanf("%d", &p);

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

completed[i] = 0;

printf("Enter the no of resources : ");

scanf("%d", &r);

printf("Enter the Max Matrix for each process : \n");

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

{

printf("For process %d : ", i + 1);

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

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

}

printf("Enter the allocation for each process : \n");

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

{

printf("For process %d : ",i + 1);

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

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

}

printf("Enter the Available Resources : ");

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

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

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

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

need[i][j] = Max[i][j] - alloc[i][j];

do

{

printf("\nMax matrix:\tAllocation matrix:\n");

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

{

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

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

printf("\t\t");

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

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

printf("\n");

}

process = -1;

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

{

if(completed[i] == 0)

{

process = i ;

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

{

if(avail[j] < need[i][j])

{

process = -1;

break;

}

}

}

if(process != -1)

break;

}

if(process != -1)

{

printf("\nProcess %d runs to completion!", process + 1);

safeSequence[count] = process + 1;

count++;

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

{

avail[j] += alloc[process][j];

alloc[process][j] = 0;

Max[process][j] = 0;

completed[process] = 1;

}

}

}while(count != p && process != -1);

if(count == p)

{

printf("\nThe system is in a safe state!!\n");

printf("Safe Sequence : < ");

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

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

printf(">\n");

}

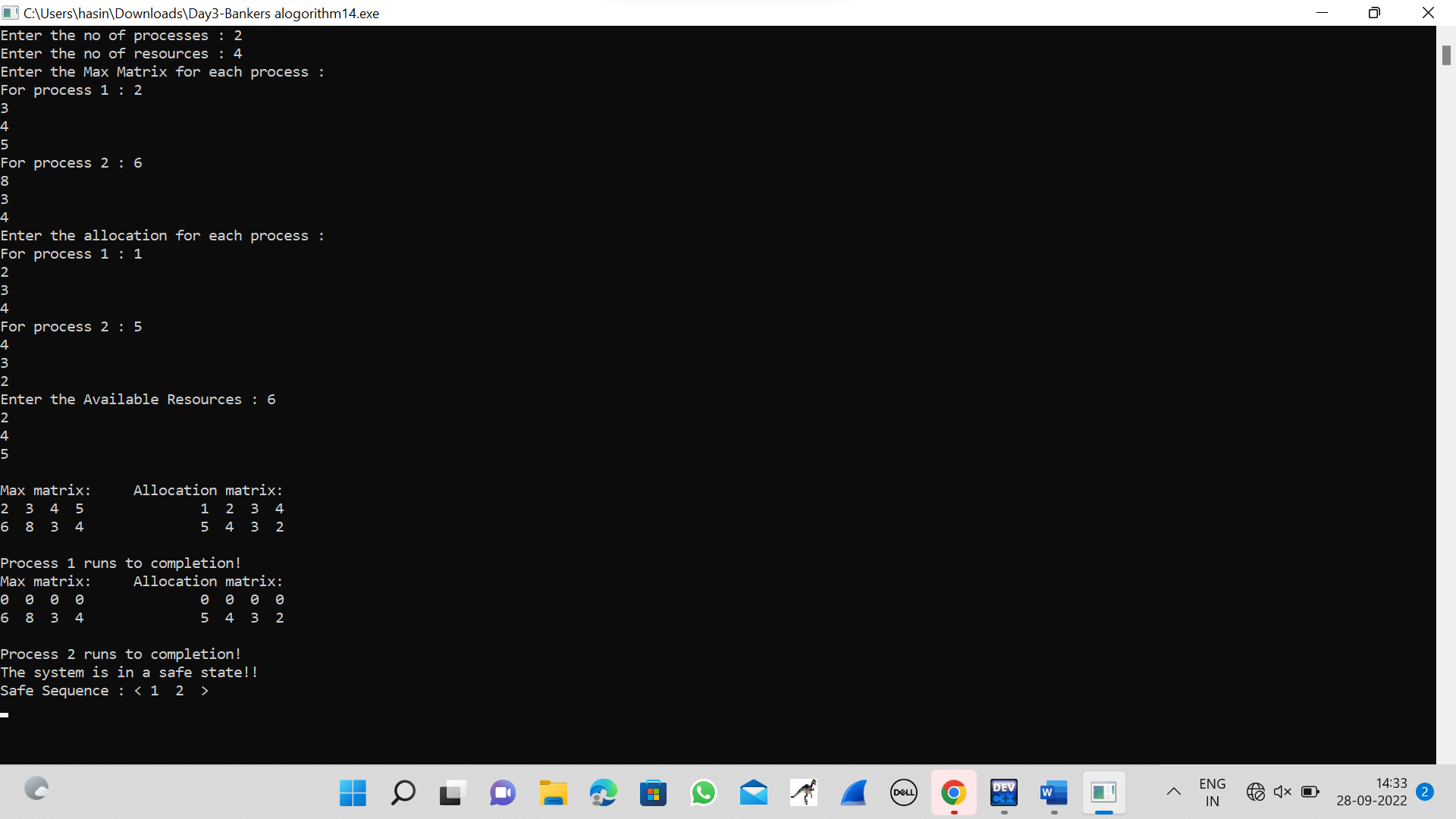
else

printf("\nThe system is in an unsafe state!!");

getch();

}

**OUTPUT**

****

**5.PRODUCER-CONSUMER PROBLEM USING SEMAPHORES**

#include <stdio.h>

#include <stdlib.h>

int mutex = 1;

int full = 0;

int empty = 10, x = 0;

void producer()

{

--mutex;

++full;

--empty;

x++;

printf("\nProducer produces"

"item %d",

x);

++mutex;

}

void consumer()

{

--mutex;

--full;

++empty;

printf("\nConsumer consumes "

"item %d",

x);

x--;

++mutex;

}

int main()

{

int n, i;

printf("\n1. Press 1 for Producer"

"\n2. Press 2 for Consumer"

"\n3. Press 3 for Exit");

for (i = 1; i > 0; i++) {

printf("\nEnter your choice:");

scanf("%d", &n);

switch (n) {

case 1:

if ((mutex == 1)

&& (empty != 0)) {

producer();

}

else {

printf("Buffer is full!");

}

break;

case 2:

if ((mutex == 1)

&& (full != 0)) {

consumer();

}

else {

printf("Buffer is empty!");

}

break;

case 3:

exit(0);

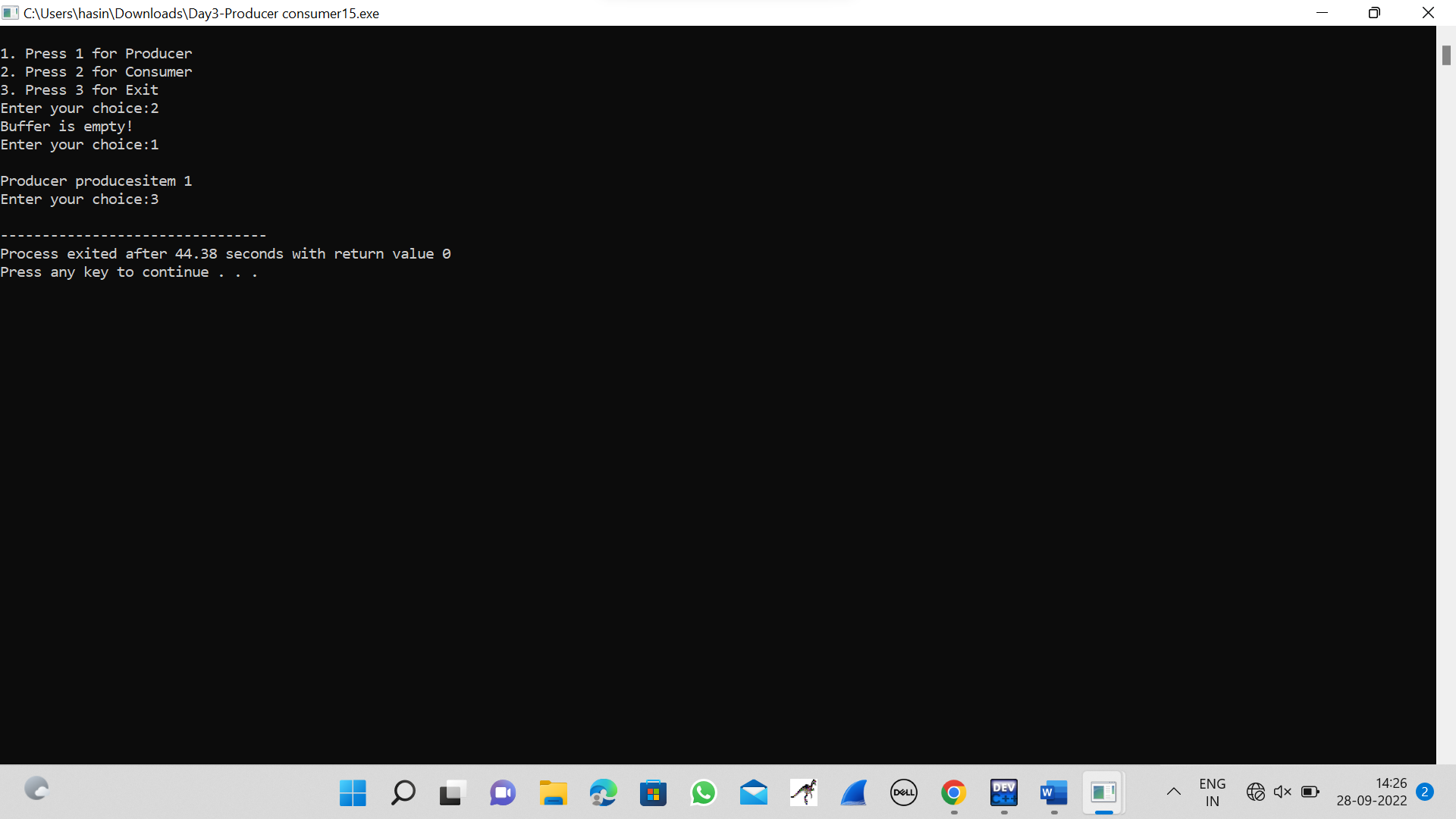
break;

}

}

}

**OUTPUT**

****