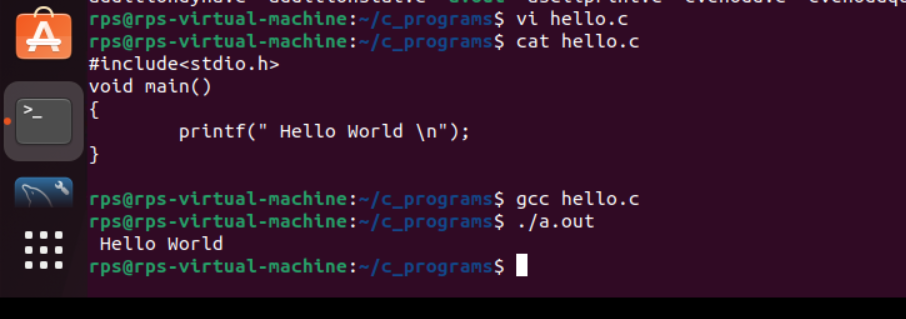
**C - Programming : All Programs\_**

**Name : YASWANTH ADAPAKALA**

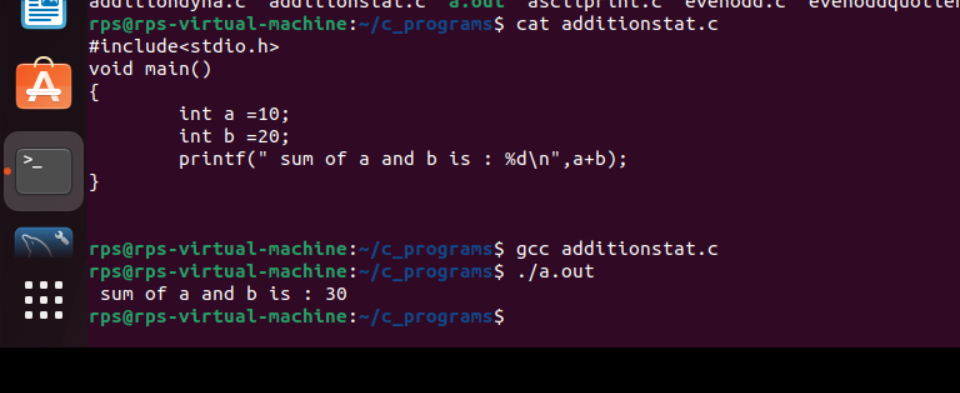
**Batch : Linux Device Driver Training**

**Trainer : Satinnder**

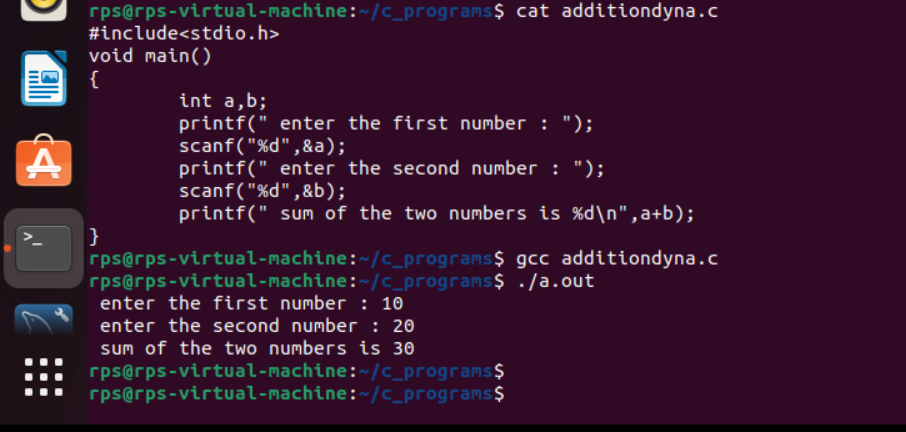
**Task 1 : Hello World Program**

****

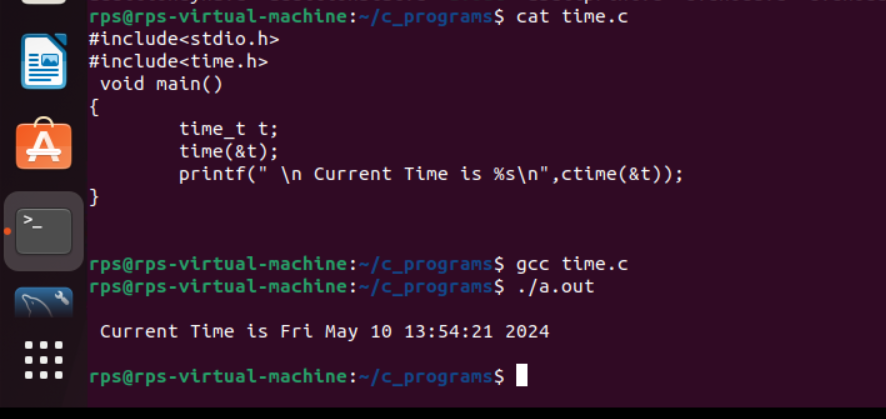
**Task 2 : Static addition**

****

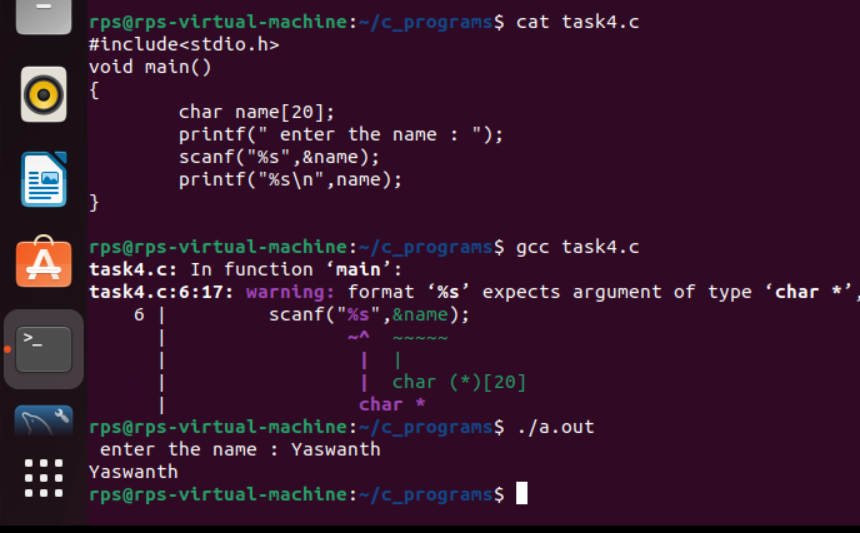
**Task 3 : Dynamic addition**

****

**Task 4 : Print date and time**

****

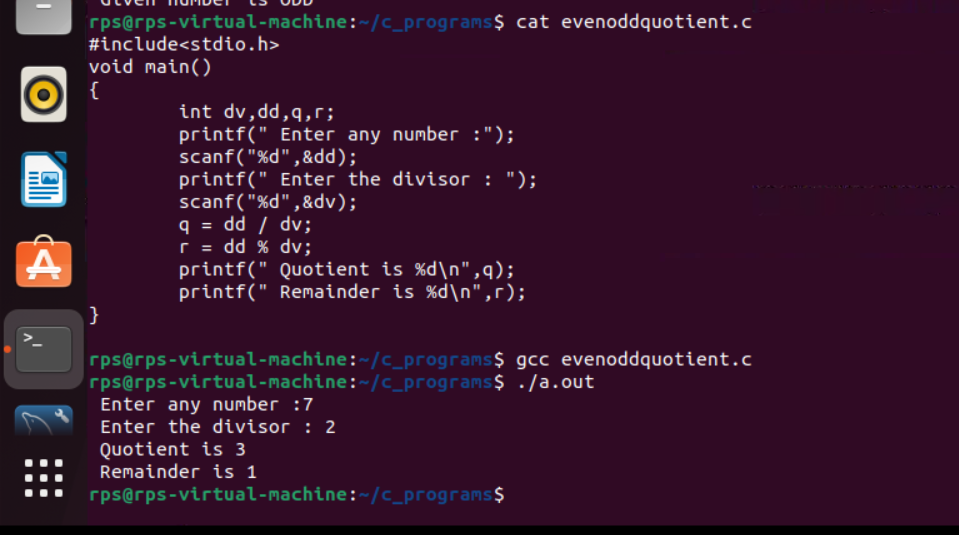
**Task 5 : Printing the name given by the user**

****

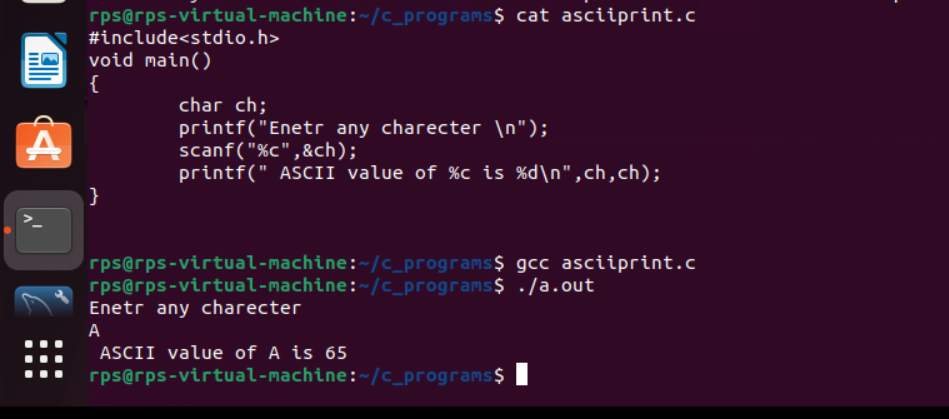
**Task 6 : Check even or odd**

****

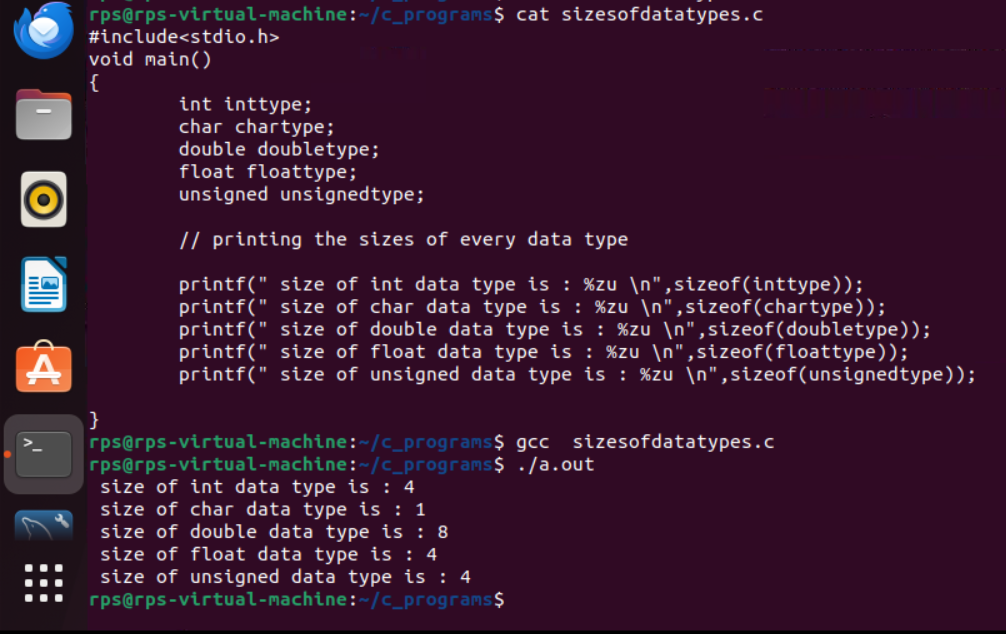
**Task 7 : Printing Quotient and Remainder**

****

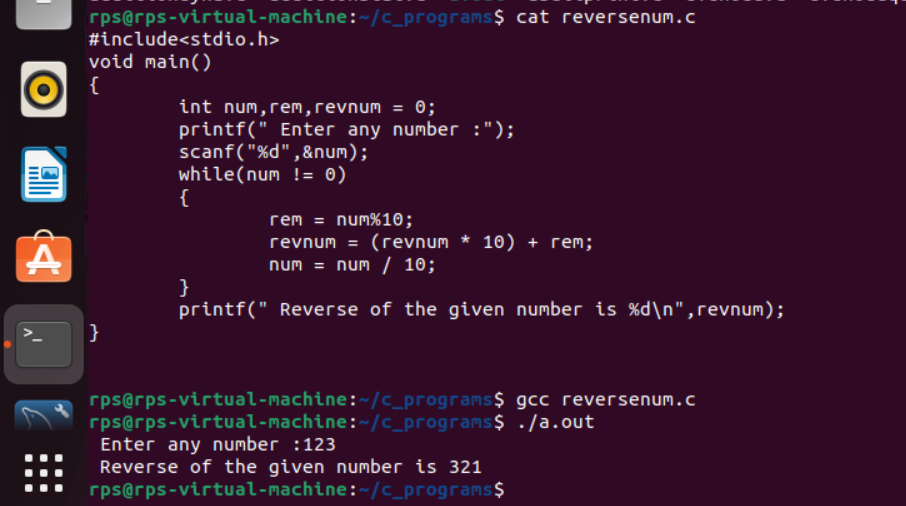
**Task 8 : Printing ASCII value of the given character**

****

**Task 9 : sizes of data types**



**Task 10 : Reverse of a number**

****

**Tasks on Conditional loops :\_**

**Task 1 : Star pattern printing**

**Code :**

#include<stdio.h>

void main()

{

int i,j,ns;

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

scanf("%d",&ns);

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

{

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

{

printf("\* ");

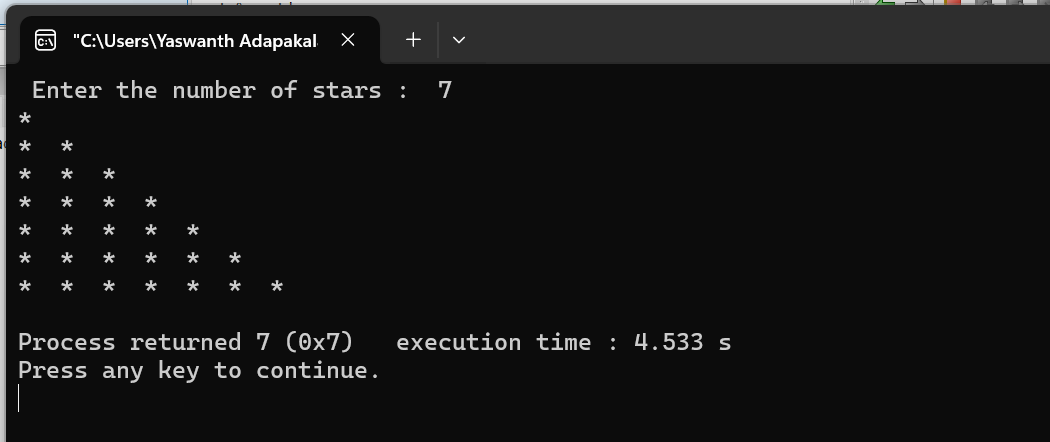
}

printf("\n");

}

}

**Output :**



**Task 2 : Fibonacci Series**

**Code :**

#include<stdio.h>

int main()

{

int num1=0,num2=1,num3,i,num;

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

scanf("%d",&num);

printf(" Required Fibonacci series is : " );

printf("%d %d ",num1,num2);

for(i=2;i<num;i++)

{

num3=num1+num2;

printf("%d ",num3);

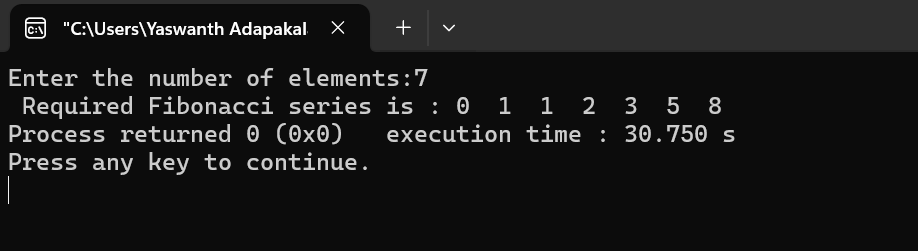
num1=num2;

num2=num3;

}

}

**Output :**



**Task 3 : Factorial of a number**

**Code :**

#include<stdio.h>

void main()

{

int i,fact=1,num;

printf("Enter any number : ");

scanf("%d",&num);

for(i=1;i<=num;i++)

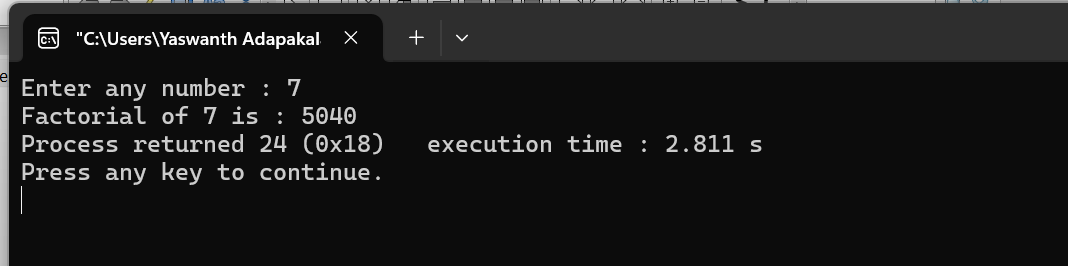
{

fact=fact\*i;

}

printf("Factorial of %d is : %d",num,fact);

}

**Output :** 

**Task 4 : while loop**

**Code :**

#include<stdio.h>

void main()

{

char username[20], domain[30];

int ch=1;

while(ch==1)

{

printf(" Enter user name : ");

scanf("%s",&username);

printf("\nEnter Domain name : ");

scanf("%s",&domain);

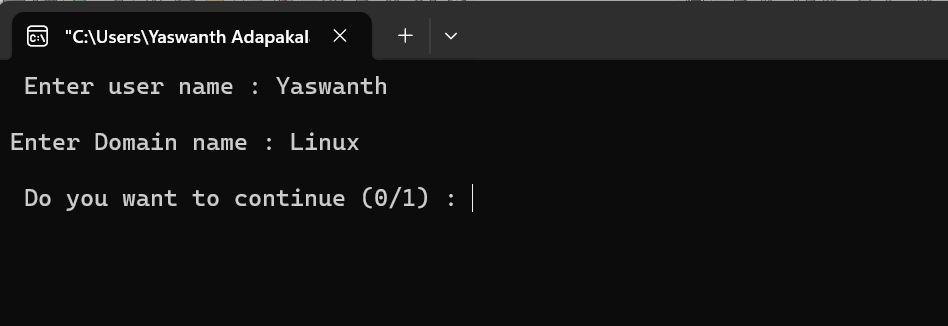
printf("\n Do you want to continue (0/1) : ");

scanf("%d",&ch);

}

}

**Output :**



**Task 5 : do-while loop**

**Code :**

#include<stdio.h>

void main()

{

char username[20], domain[30];

int ch=1;

do{

printf(" Enter user name : ");

scanf("%s",&username);

printf("\nEnter Domain name : ");

scanf("%s",&domain);

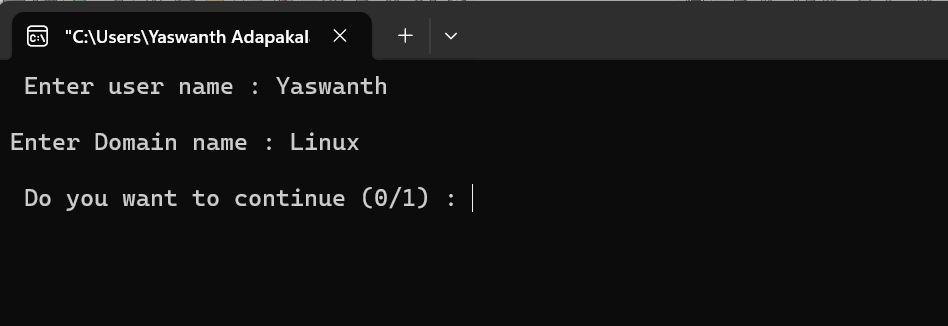
printf("\n Do you want to continue (0/1) : ");

scanf("%d",&ch);

}while(ch==1);

}

**Output :**



**Task 6 : printing matrices**

**Code :**

#include<stdio.h>

#include<stdlib.h>

void main()

{

int a[2][2],b[2][2],res[2][2],i,j,k;

system("cls");

printf("enter the first matrix elements : \n");

for(i=0;i<2;i++)

{

for(j=0;j<2;j++)

{

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

}

}

printf("enter the second matrix elements : \n");

for(i=0;i<2;i++)

{

for(j=0;j<2;j++)

{

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

}

}

// Printing the matrices -------------------

printf("\n printing the first matrix elements : \n");

for(i=0;i<2;i++)

{

for(j=0;j<2;j++)

{

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

}

printf("\n");

}

printf("\n printing the second matrix elements : \n");

for(i=0;i<2;i++)

{

for(j=0;j<2;j++)

{

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

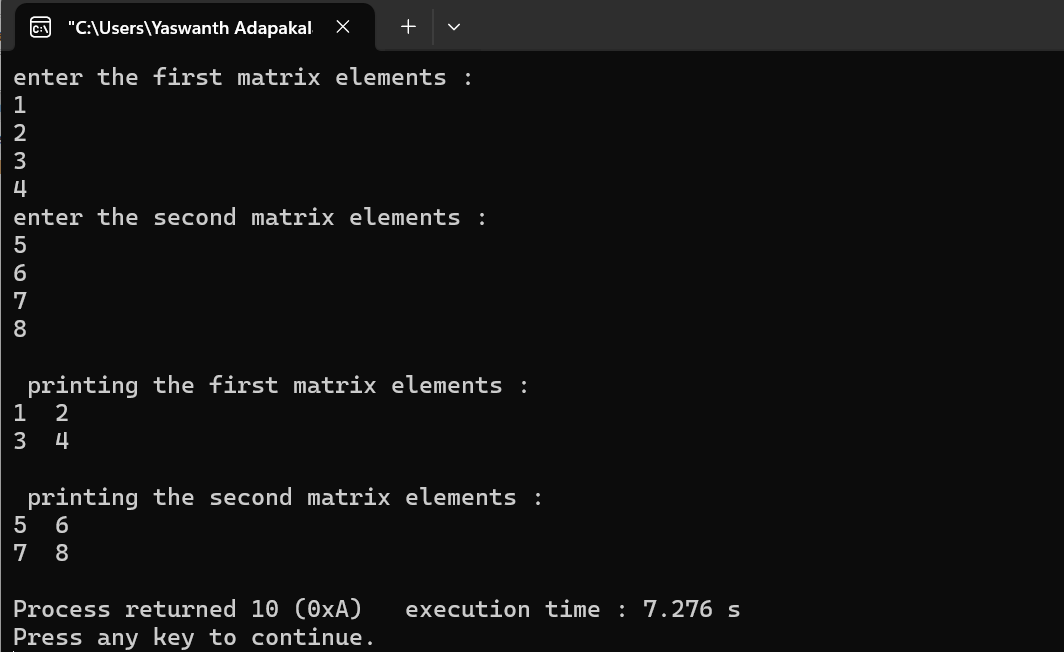
}

printf("\n");

}

}

**Output :**



**Task 7 : Matrix multiplication**

**Code :**

#include<stdio.h>

#include<stdlib.h>

void main()

{

int a[2][2],b[2][2],res[2][2],i,j,k;

system("cls");

printf("enter the first matrix elements : \n");

for(i=0;i<2;i++)

{

for(j=0;j<2;j++)

{

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

}

}

printf("enter the second matrix elements : \n");

for(i=0;i<2;i++)

{

for(j=0;j<2;j++)

{

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

}

}

printf("Multiply of the matrix : \n");

for(i=0;i<2;i++)

{

for(j=0;j<2;j++)

{

res[i][j]=0;

for(k=0;k<2;k++)

{

res[i][j]+=a[i][k]\*b[k][j];

}

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

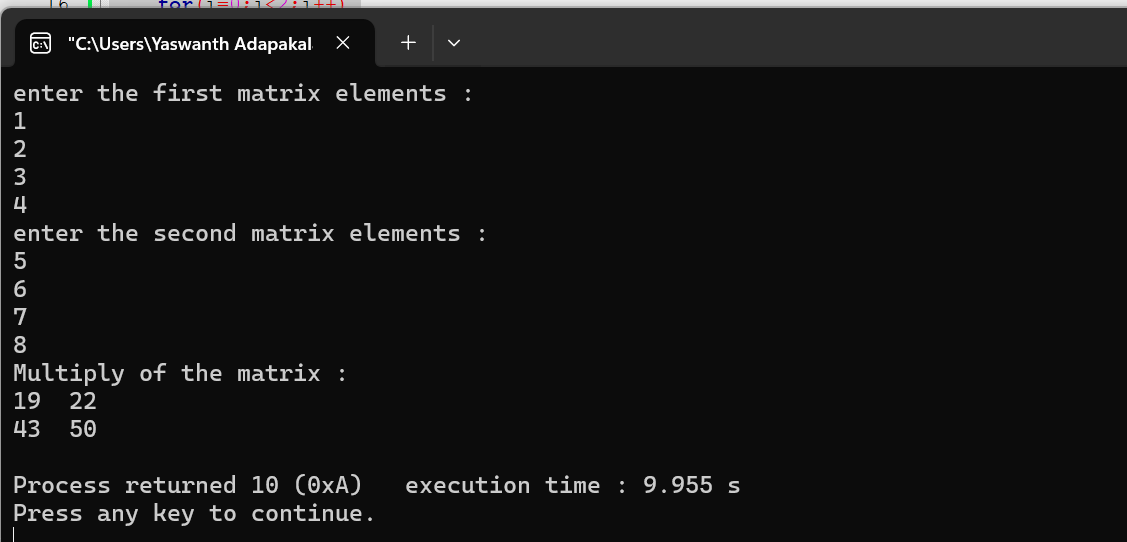
}

printf("\n");

}

}

**Output :**



**Tasks on Arrays : \_**

**Task 1 :**

**Code :**

#include<stdio.h>

void main()

{

int a[20],i;

printf(" Enter the elements of the array of 20 : \n ");

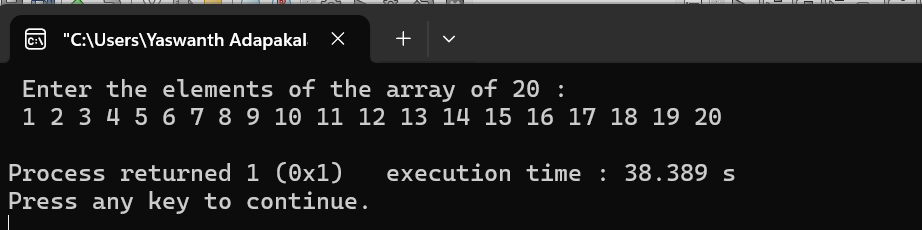
for(i=0;i<20;i++)

{

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

}

}

**Output :**

**Task 2 :**

**Code :**

#include<stdio.h>

void main()

{

int a[20],i;

printf(" Enter the elements of the array of 20 : \n ");

for(i=0;i<20;i++)

{

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

}

printf("\n printing the elements of the array with index values :\n");

for(i=0;i<20;i++)

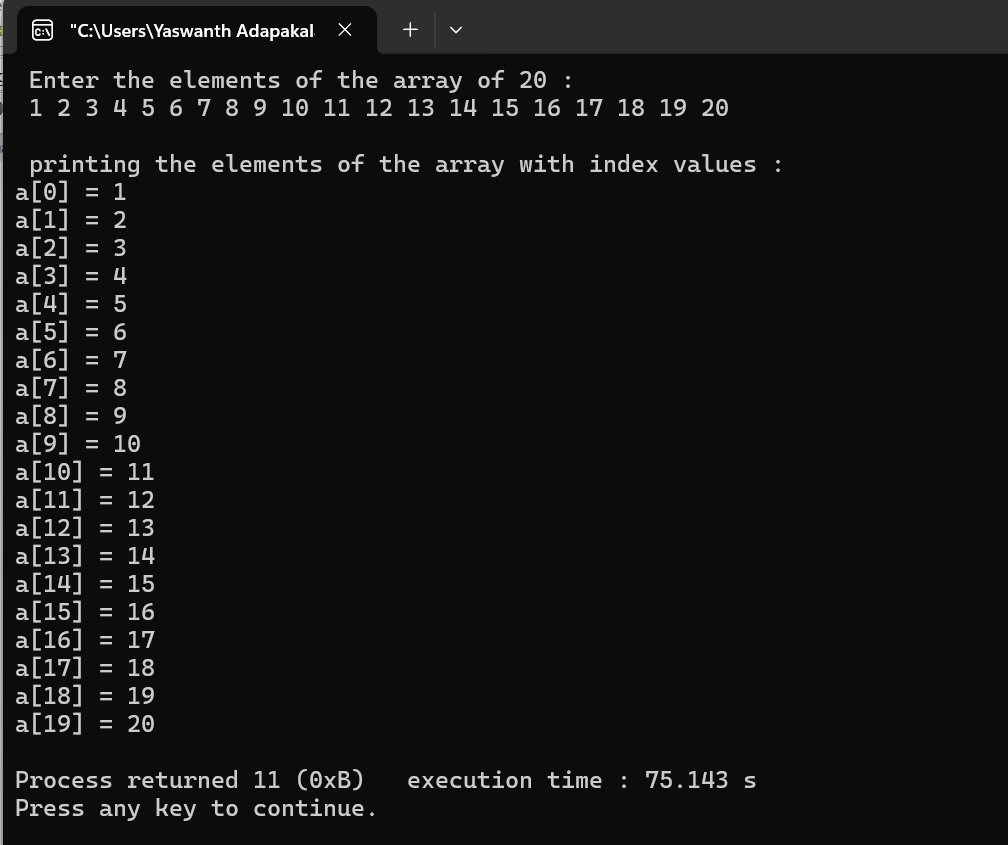
{

printf("a[%d] = %d\n",i,a[i]);

}

}

**Output :**



**Task 3 :**

**Code :**

#include<stdio.h>

void main()

{

int a[20],i,j;

printf(" Enter the elements of the array of 20 : \n ");

for(i=0;i<20;i++)

{

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

}

printf("\n Enter the index of value which is to be deleted : ");

scanf("%d",&j);

a[j]=0;

printf("\n printing the elements of the array after deleting the element :\n");

for(i=0;i<20;i++)

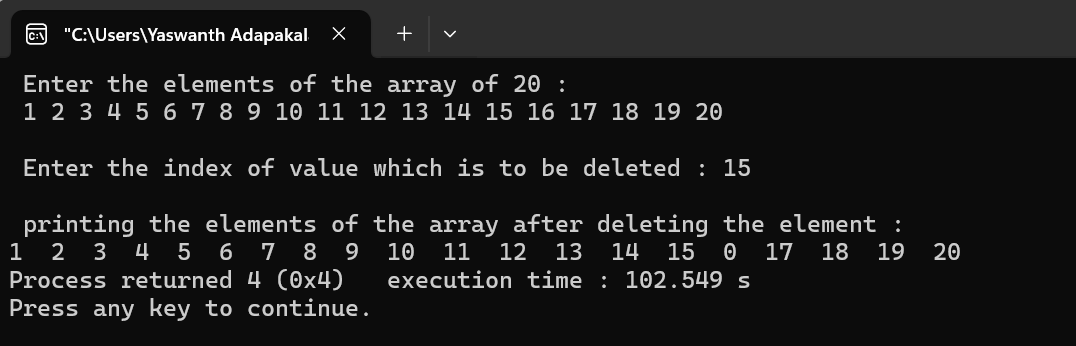
{

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

}

}

**Output :**



**Task 4 :**

**Code :**

#include<stdio.h>

void main()

{

int a[20],i,j;

printf(" Enter the elements of the array of 20 : \n ");

for(i=0;i<20;i++)

{

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

}

printf("\n Printing index values of duplicate elements : \n");

for (i=0;i<20;i++)

{

for(j=i+1;j<20;j++)

{

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

{

printf("%d and %d \n",i,j);

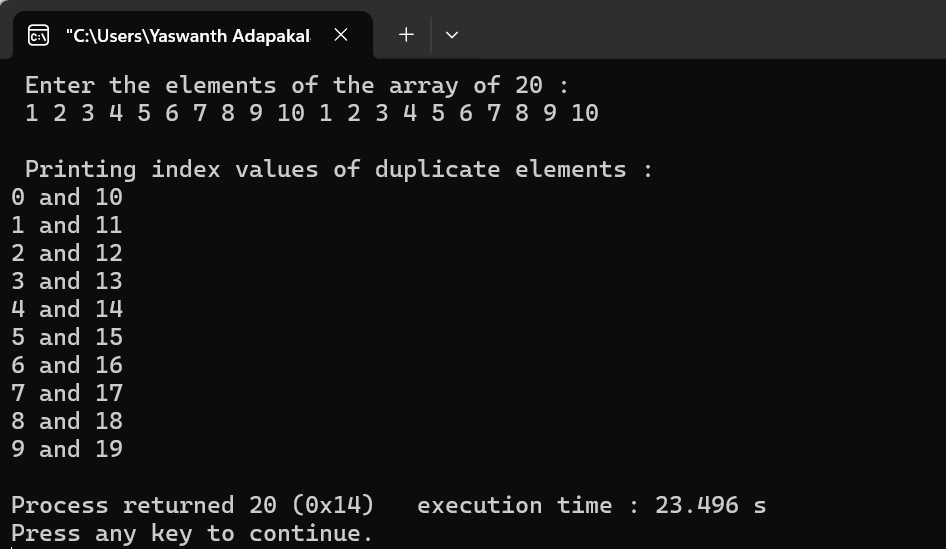
}

}

}

}

**Output :**

****

**Task 5 :**

**Code :**

#include<stdio.h>

void main()

{

int a[20],i,num,flag=0;

printf(" Enter the elements of the array of 20 : \n ");

for(i=0;i<20;i++)

{

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

}

printf("\n Enter the element which is to be searched :\n");

scanf("%d",&num);

for(i=0;i<20;i++)

{

if(a[i]==num)

{

printf(" Given number is found at index value : %d \n",i);

flag++;

}

}

if(flag==0)

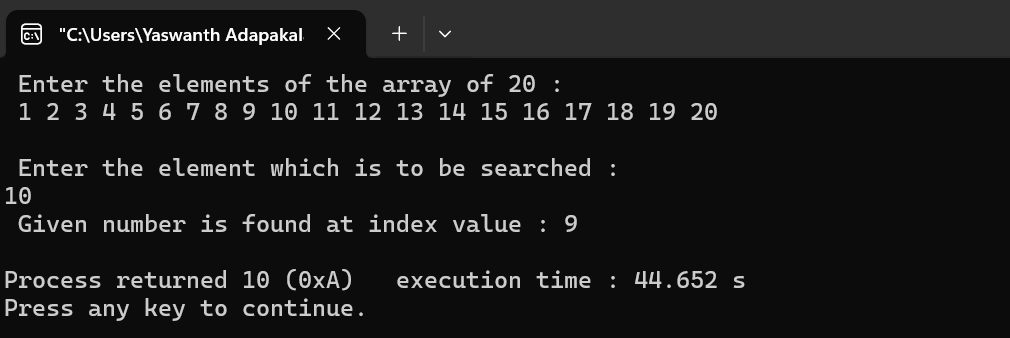
{

printf(" Element NOT FOUND ");

}

}

**Output :**



**Tasks on File Handling functions :\_**

**Task 1 :**

**Code :**

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

void main ()

{

int n;

FILE \*fptr;

fptr = fopen ("C:\\Users\\Yaswanth Adapakala\\Desktop\\Linux\\Codeblocks\_programs\\sampletext.txt", "w");

if (fptr == NULL)

{

printf ("Error!!!!!");

exit(0);

}

printf ("Enter number::");

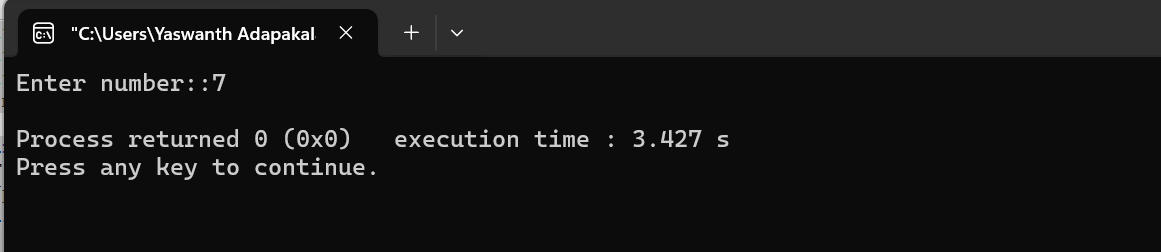
scanf ("%d", &n);

fprintf (fptr, "%d", n);

fclose (fptr);

}

**Output :**



**Task 2 :**

**Code :**

#include <stdio.h>

#include <conio.h>

#include <string.h>

struct emp

{

char name[30];

int age;

float salary;

};

void main()

{

struct emp p1;

strcpy (p1.name, "myname");

p1.age = 27;

p1.salary = 10000;

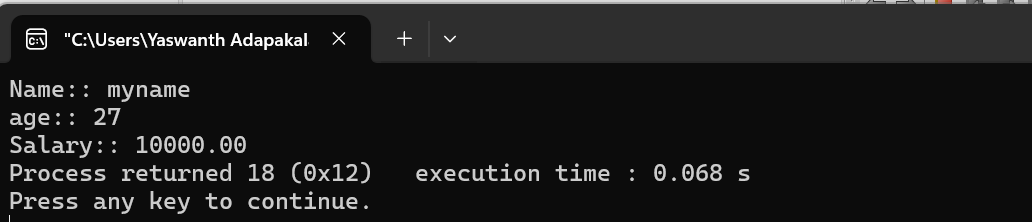
printf("Name:: %s", p1.name);

printf("\nage:: %d", p1.age);

printf("\nSalary:: %.2f", p1.salary);

}

**Output :**



**Task 3 :**

**Code :**

#include <stdio.h>

#include <conio.h>

#include <string.h>

struct emp

{

char name[30];

int age;

char department[20];

};

void main()

{

struct emp p1;

FILE \*fptr;

fptr = fopen ("C:\\Users\\Yaswanth Adapakala\\Desktop\\Linux\\Codeblocks\_programs\\empdetails.txt", "a");

printf(" Enter Employee name : ")

scanf("%s",&p1.name);

printf(" \nEnter Employee Age : ");

scanf("%d",&p1.age);

printf(" \nEnter Employee Department : ");

scanf("%s",&p1.department);

fprintf (fptr, "Employee name : %s\n", p1.name);

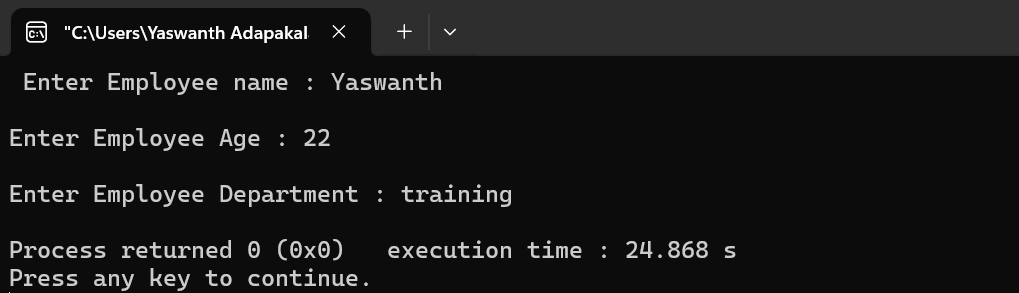
fprintf (fptr, "Employee Age : %d\n", p1.age);

fprintf (fptr, "Employee Department : %s\n", p1.department);

fclose (fptr);

}

**Output :**



**Task 4 :**

**Code :**

#include <stdio.h> ///for input output functions like printf, scanf

#include <stdlib.h>

#include <conio.h>

#include <windows.h> ///for windows related functions (not important)

#include <string.h> ///string operations

/\*\* List of Global Variable \*/

COORD coord = {0,0}; /// top-left corner of window

/\*\*

function : gotoxy

@param input: x and y coordinates

@param output: moves the cursor in specified position of console

\*/

void gotoxy(int x,int y)

{

coord.X = x;

coord.Y = y;

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE),coord);

}

/\*\* Main function started \*/

int main()

{

FILE \*fp, \*ft; /// file pointers

char another, choice;

/\*\* structure that represent a employee \*/

struct emp

{

char name[40]; ///name of employee

int age; /// age of employee

float bs; /// basic salary of employee

};

struct emp e; /// structure variable creation

char empname[40]; /// string to store name of the employee

long int recsize; /// size of each record of employee

/\*\* open the file in binary read and write mode

\* if the file EMP.DAT already exists then it open that file in read write mode

\* if the file doesn't exit it simply create a new copy

\*/

fp = fopen("EMP.DAT","rb+");

if(fp == NULL)

{

fp = fopen("EMP.DAT","wb+");

if(fp == NULL)

{

printf("Connot open file");

exit(1);

}

}

/// sizeo of each record i.e. size of structure variable e

recsize = sizeof(e);

/// infinite loop continues untile the break statement encounter

while(1)

{

system("cls"); ///clear the console window

gotoxy(30,10); /// move the cursor to postion 30, 10 from top-left corner

printf("1. Add Record"); /// option for add record

gotoxy(30,12);

printf("2. List Records"); /// option for showing existing record

gotoxy(30,14);

printf("3. Modify Records"); /// option for editing record

gotoxy(30,16);

printf("4. Delete Records"); /// option for deleting record

gotoxy(30,18);

printf("5. Exit"); /// exit from the program

gotoxy(30,20);

printf("Your Choice: "); /// enter the choice 1, 2, 3, 4, 5

fflush(stdin); /// flush the input buffer

choice = getche(); /// get the input from keyboard

switch(choice)

{

case '1': /// if user press 1

system("cls");

fseek(fp,0,SEEK\_END); /// search the file and move cursor to end of the file

/// here 0 indicates moving 0 distance from the end of the file

another = 'y';

while(another == 'y') /// if user want to add another record

{

printf("\nEnter name: ");

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

printf("\nEnter age: ");

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

printf("\nEnter basic salary: ");

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

fwrite(&e,recsize,1,fp); /// write the record in the file

printf("\nAdd another record(y/n) ");

fflush(stdin);

another = getche();

}

break;

case '2':

system("cls");

rewind(fp); ///this moves file cursor to start of the file

while(fread(&e,recsize,1,fp)==1) /// read the file and fetch the record one record per fetch

{

printf("\n%s %d %.2f",e.name,e.age,e.bs); /// print the name, age and basic salary

}

getch();

break;

case '3': /// if user press 3 then do editing existing record

system("cls");

another = 'y';

while(another == 'y')

{

printf("Enter the employee name to modify: ");

scanf("%s", empname);

rewind(fp);

while(fread(&e,recsize,1,fp)==1) /// fetch all record from file

{

if(strcmp(e.name,empname) == 0) ///if entered name matches with that in file

{

printf("\nEnter new name,age and bs: ");

scanf("%s%d%f",e.name,&e.age,&e.bs);

fseek(fp,-recsize,SEEK\_CUR); /// move the cursor 1 step back from current position

fwrite(&e,recsize,1,fp); /// override the record

break;

}

}

printf("\nModify another record(y/n)");

fflush(stdin);

another = getche();

}

break;

case '4':

system("cls");

another = 'y';

while(another == 'y')

{

printf("\nEnter name of employee to delete: ");

scanf("%s",empname);

ft = fopen("Temp.dat","wb"); /// create a intermediate file for temporary storage

rewind(fp); /// move record to starting of file

while(fread(&e,recsize,1,fp) == 1) /// read all records from file

{

if(strcmp(e.name,empname) != 0) /// if the entered record match

{

fwrite(&e,recsize,1,ft); /// move all records except the one that is to be deleted to temp file

}

}

fclose(fp);

fclose(ft);

remove("EMP.DAT"); /// remove the orginal file

rename("Temp.dat","EMP.DAT"); /// rename the temp file to original file name

fp = fopen("EMP.DAT", "rb+");

printf("Delete another record(y/n)");

fflush(stdin);

another = getche();

}

break;

case '5':

fclose(fp); /// close the file

exit(0); /// exit from the program

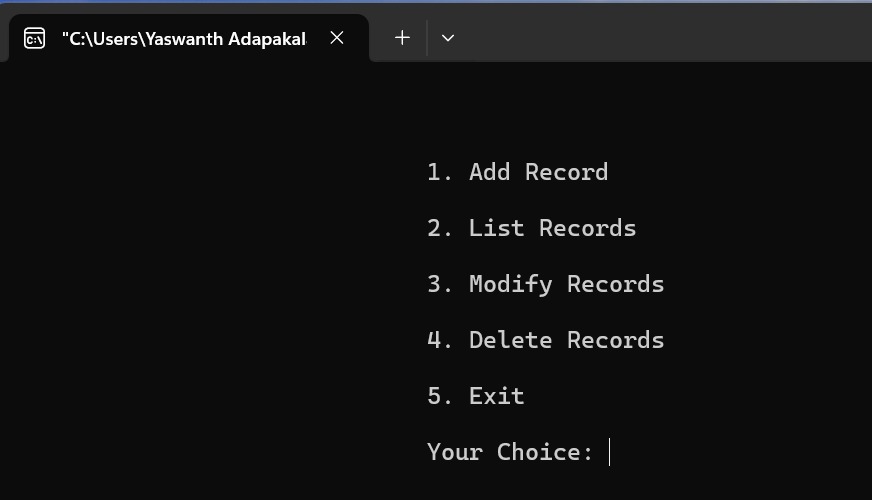
}

}

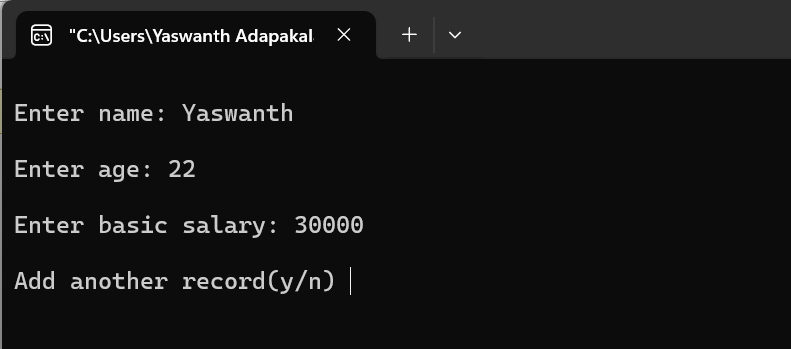
return 0;

}

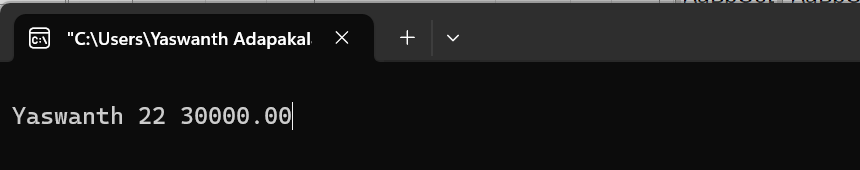
**Output 1 :**



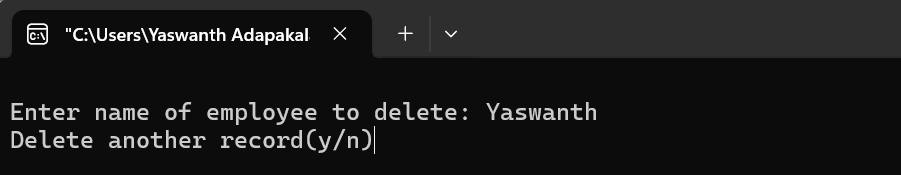
**Output 2 :**



**Output 3 :**



**Output 4 :**



**Task 5 :**

**Code :**

#include <stdio.h>

#include <string.h>

void create();

void view();

void edit();

void delete();

void choice();

void choice(){

int choic;

printf("\nWELCOME TO OMG COMPANY\n");

start:

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

printf("Press 1 to Create a new Data\n");

printf("Press 2 to View the Data\n");

printf("Press 3 to edit the Data\n");

printf("Press 4 to delete the Data\n");

printf("Press 5 to exit the Program...!\n");

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

int choice;

scanf("%d",&choice);

if(choice==5){

printf("Exiting the Program...\n");

return ;

}

switch (choice)

{

case 1:

create();

printf("Do you wish to continue? 1 for Yes 0 for No\n");

scanf("%d",&choic);

if(choic==1){

goto start;

}

else if(choic==0){

printf(" exiting the program......!\n");

}

break;

case 2:

view();

printf("Do you wish to continue? 1 for Yes 0 for No\n");

scanf("%d",&choic);

if(choic==1){

goto start;

}

else if(choic==0){

printf(" exiting the program......!\n");

}

break;

case 3:

edit();

printf("Do you wish to continue? 1 for Yes 0 for No\n");

scanf("%d",&choic);

if(choic==1){

goto start;

}

else if(choic==0){

printf(" exiting the program......!\n");

}

break;

case 4:

delete();

printf("Do you wish to continue? 1 for Yes 0 for No\n");

scanf("%d",&choic);

if(choic==1){

goto start;

}

else if(choic==0){

printf(" exiting the program......!\n");

}

break;

default:

break;

}

}

struct emp

{

int id;

char name[30];

int age;

char branch[20];

char designation[20];

}emp;

void create(){

struct emp p1;

FILE \*fp=fopen("C:\\Users\\Yaswanth Adapakala\\Desktop\\Linux\\Codeblocks\_programs\\empdetails.txt", "a");

if(fp==NULL){

printf("Error opening File");

return ;

}

printf("\nEnter Employee Id:");

scanf("%d",&p1.id);

// fflush(stdin);

printf("\nEnter Employee name: ");

scanf("%s", p1.name);

strcpy (p1.name, p1.name);

printf("\nEnter Employee Age: ");

scanf("%d", &p1.age);

printf("\nEnter Employee Branch:");

scanf("%s", p1.branch);

printf("\nEnter Employee Designation:");

scanf("%s", p1.designation);

fwrite(&p1,sizeof(emp),1,fp);

printf("Data Uploaded Successfully\n");

fclose(fp);

}

void view(){

struct emp p1;

FILE \*ptr;

ptr=fopen("C:\\Users\\Yaswanth Adapakala\\Desktop\\Linux\\Codeblocks\_programs\\empdetails.txt", "r");

if(ptr==NULL){

printf("Failed to open File\n");

return;

}

while(fread(&p1,sizeof(emp),1,ptr)){

printf("%d %s %d %s %s\n",p1.id,p1.name,p1.age,p1.branch,p1.designation);

}

fclose(ptr);

}

void edit(){

struct emp p1;

FILE \*ptr, \*ptr1;

int c = 0;

int a;

ptr = fopen("C:\\Users\\Yaswanth Adapakala\\Desktop\\Linux\\Codeblocks\_programs\\empdetails.txt", "r");

if (ptr == NULL) {

printf("Error opening file for reading\n");

return ;

}

ptr1 = fopen("C:\\Users\\Yaswanth Adapakala\\Desktop\\Linux\\Codeblocks\_programs\\empdetails.txt", "w");

if (ptr1 == NULL) {

printf("Error opening file for writing\n");

fclose(ptr);

return ;

}

printf("Enter the Employee Id to Modify the Data:\n");

scanf("%d", &a);

while (fread(&p1, sizeof(struct emp), 1, ptr)) {

if (a == p1.id) {

c = 1;

printf("Enter New Employee name: \n");

scanf("%s", p1.name);

printf("Enter New Employee Age: \n");

scanf("%d", &p1.age);

printf("Enter New Employee Branch:\n");

scanf("%s", p1.branch);

printf("Enter New Employee Designation:\n");

scanf("%s", p1.designation);

}

fwrite(&p1, sizeof(struct emp), 1, ptr1);

}

fclose(ptr);

fclose(ptr1);

if (c == 0) {

printf("Employee Not found\n");

} else {

ptr1 = fopen("C:\\Users\\Yaswanth Adapakala\\Desktop\\Linux\\Codeblocks\_programs\\empdetails.txt", "r");

ptr = fopen("C:\\Users\\Yaswanth Adapakala\\Desktop\\Linux\\Codeblocks\_programs\\empdetails.txt", "w");

if (ptr == NULL) {

printf("Error opening file for writing\n");

fclose(ptr1);

return ;

}

if (ptr1 == NULL) {

printf("Error opening file for reading\n");

fclose(ptr);

return ;

}

while (fread(&p1, sizeof(struct emp), 1, ptr1)) {

fwrite(&p1, sizeof(struct emp), 1, ptr);

}

fclose(ptr);

fclose(ptr1);

remove("C:\\Users\\Yaswanth Adapakala\\Desktop\\Linux\\Codeblocks\_programs\\empdetails.txt ");

}

}

void delete(){

FILE \*ptr, \*ptr1;

struct emp p1;

ptr = fopen("C:\\Users\\Yaswanth Adapakala\\Desktop\\Linux\\Codeblocks\_programs\\empdetails.txt", "r");

ptr1 = fopen("C:\\Users\\Yaswanth Adapakala\\Desktop\\Linux\\Codeblocks\_programs\\empdetails.txt", "w");

if(ptr == NULL){

printf("Error opening files for deleting\n");

fclose(ptr);

return;

}

if(ptr1==NULL){

printf("Error opening files for deleting\n");

fclose(ptr1);

return ;

}

printf("Enter the employee id to delete\n");

int a;

scanf("%d", &a);

int c = 0;

while (fread(&p1, sizeof(struct emp), 1, ptr)){

printf("%d",p1.id);

if(a == p1.id){

c = 1;

} else {

fwrite(&p1, sizeof(struct emp), 1, ptr1);

}

}

fclose(ptr1);

fclose(ptr);

if(c == 1){

ptr = fopen("C:\\Users\\Yaswanth Adapakala\\Desktop\\Linux\\Codeblocks\_programs\\empdetails.txt", "r");

ptr1 = fopen("C:\\Users\\Yaswanth Adapakala\\Desktop\\Linux\\Codeblocks\_programs\\empdetails.txt", "w");

if(ptr == NULL || ptr1 == NULL){

printf("Failed to open the file\n");

if(ptr) fclose(ptr);

if(ptr1) fclose(ptr1);

return;

}

while(fread(&p1, sizeof(struct emp), 1, ptr)){

fwrite(&p1, sizeof(struct emp), 1, ptr1);

}

fclose(ptr1);

fclose(ptr);

remove("C:\\Users\\Yaswanth Adapakala\\Desktop\\Linux\\Codeblocks\_programs\\empdetails.txt");

printf("Deleted the employee successfully\n");

} else {

printf("Record not found\n");

}

}

int main()

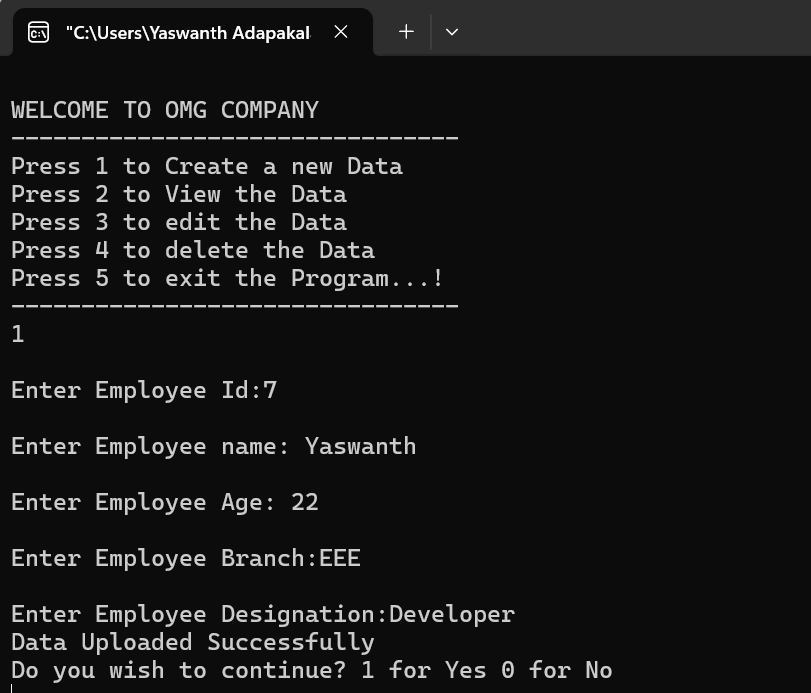
{

choice();

return 0;

}

**Output :**



**Task 6 :**

**Code :**

// C program to print the month by month

// calendar for the given year

#include <stdio.h>

// Function that returns the index of the

// day for date DD/MM/YYYY

int dayNumber(int day, int month, int year)

{

static int t[] = { 0, 3, 2, 5, 0, 3,

5, 1, 4, 6, 2, 4 };

year -= month < 3;

return (year + year / 4 - year / 100 + year / 400 + t[month - 1] + day) % 7;

}

// Function that returns the name of the

// month for the given month Number

// January - 0, February - 1 and so on

char\* getMonthName(int monthNumber)

{

char\* month;

switch (monthNumber) {

case 0:

month = "January";

break;

case 1:

month = "February";

break;

case 2:

month = "March";

break;

case 3:

month = "April";

break;

case 4:

month = "May";

break;

case 5:

month = "June";

break;

case 6:

month = "July";

break;

case 7:

month = "August";

break;

case 8:

month = "September";

break;

case 9:

month = "October";

break;

case 10:

month = "November";

break;

case 11:

month = "December";

break;

}

return month;

}

// Function to return the number of days

// in a month

int numberOfDays(int monthNumber, int year)

{

// January

if (monthNumber == 0)

return (31);

// February

if (monthNumber == 1) {

// If the year is leap then Feb

// has 29 days

if (year % 400 == 0 || (year % 4 == 0 && year % 100 != 0))

return (29);

else

return (28);

}

// March

if (monthNumber == 2)

return (31);

// April

if (monthNumber == 3)

return (30);

// May

if (monthNumber == 4)

return (31);

// June

if (monthNumber == 5)

return (30);

// July

if (monthNumber == 6)

return (31);

// August

if (monthNumber == 7)

return (31);

// September

if (monthNumber == 8)

return (30);

// October

if (monthNumber == 9)

return (31);

// November

if (monthNumber == 10)

return (30);

// December

if (monthNumber == 11)

return (31);

}

// Function to print the calendar of

// the given year

void printCalendar(int year)

{

printf(" Calendar - %d\n\n", year);

int days;

// Index of the day from 0 to 6

int current = dayNumber(1, 1, year);

// i for Iterate through months

// j for Iterate through days

// of the month - i

for (int i = 0; i < 12; i++) {

days = numberOfDays(i, year);

// Print the current month name

printf("\n ------------%s-------------\n",

getMonthName(i));

// Print the columns

printf(" Sun Mon Tue Wed Thu Fri Sat\n");

// Print appropriate spaces

int k;

for (k = 0; k < current; k++)

printf(" ");

for (int j = 1; j <= days; j++) {

printf("%5d", j);

if (++k > 6) {

k = 0;

printf("\n");

}

}

if (k)

printf("\n");

current = k;

}

return;

}

// Driver Code

int main()

{

int year;

printf(" Enter the year whose calender is to be printed : ");

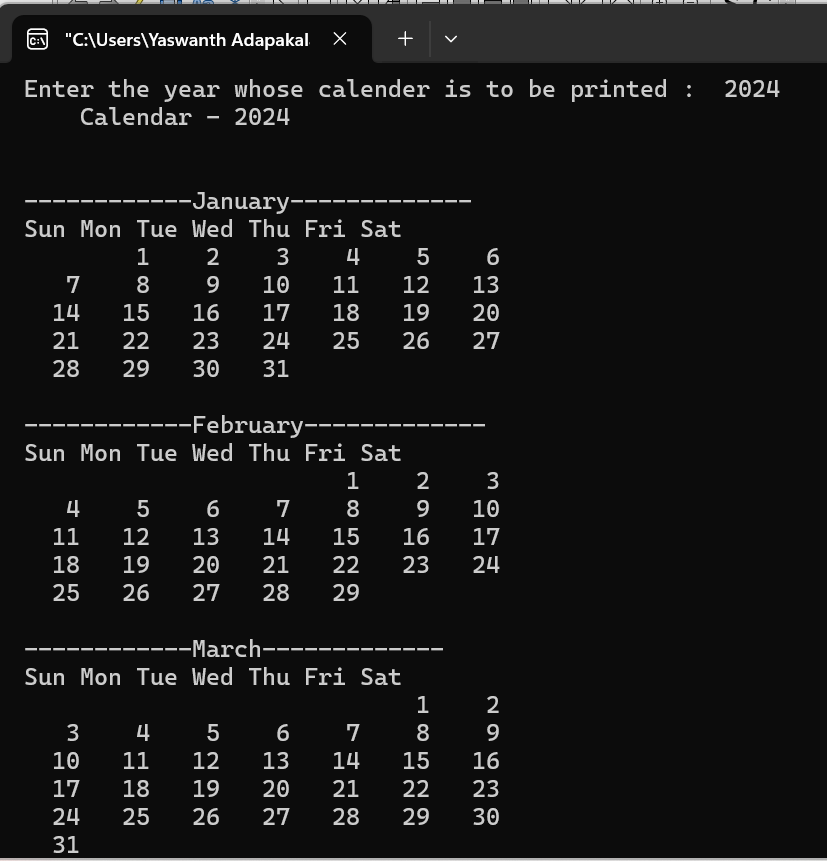
scanf("%d",&year);

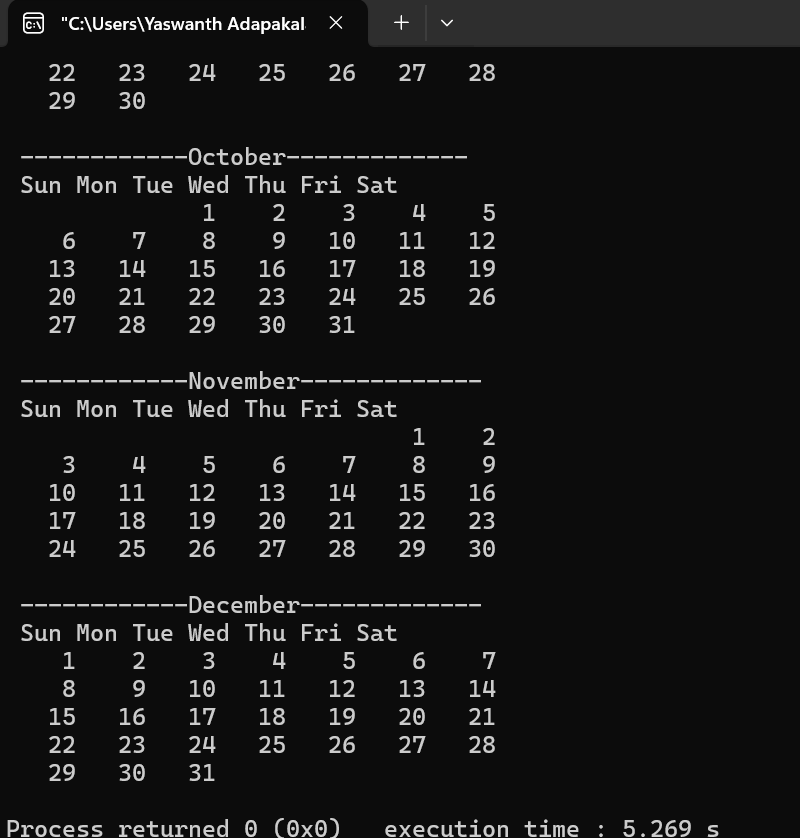
// Function Call

printCalendar(year);

return 0;

}

**Output :** 



**Task 7 :**

**Code :**

#include <stdio.h>

int dayNumber(int day, int month, int year)

{

static int t[] = { 0, 3, 2, 5, 0, 3,

5, 1, 4, 6, 2, 4 };

year -= month < 3;

return (year + year / 4 - year / 100 + year / 400 + t[month - 1] + day) % 7;

}

char\* getMonthName(int monthNumber)

{

char\* month;

switch (monthNumber) {

case 0:

month = "January";

break;

case 1:

month = "February";

break;

case 2:

month = "March";

break;

case 3:

month = "April";

break;

case 4:

month = "May";

break;

case 5:

month = "June";

break;

case 6:

month = "July";

break;

case 7:

month = "August";

break;

case 8:

month = "September";

break;

case 9:

month = "October";

break;

case 10:

month = "November";

break;

case 11:

month = "December";

break;

}

return month;

}

int numberOfDays(int monthNumber, int year)

{

if (monthNumber == 0)

return (31);

if (monthNumber == 1) {

if (year % 400 == 0 || (year % 4 == 0 && year % 100 != 0))

return (29);

else

return (28);

}

if (monthNumber == 2)

return (31);

if (monthNumber == 3)

return (30);

if (monthNumber == 4)

return (31);

if (monthNumber == 5)

return (30);

if (monthNumber == 6)

return (31);

if (monthNumber == 7)

return (31);

if (monthNumber == 8)

return (30);

if (monthNumber == 9)

return (31);

if (monthNumber == 10)

return (30);

if (monthNumber == 11)

return (31);

}

void printCalendar(int year, int i)

{

printf(" Calendar - %d\n\n", year);

int days;

int current = dayNumber(1, 1, year);

days = numberOfDays(i, year);

printf("\n ------------%s-------------\n",

getMonthName(i));

// Print the columns

printf(" Sun Mon Tue Wed Thu Fri Sat\n");

// Print appropriate spaces

int k;

for (k = 0; k < current; k++)

printf(" ");

for (int j = 1; j <= days; j++) {

printf("%5d", j);

if (++k > 6) {

k = 0;

printf("\n");

}

}

if (k)

printf("\n");

current = k;

return;

}

// Driver Code

int main()

{

int year,i;

printf(" Enter the year : ");

scanf("%d",&year);

printf("\n Enter the Month : ");

scanf("%d",&i);

i=i-1;

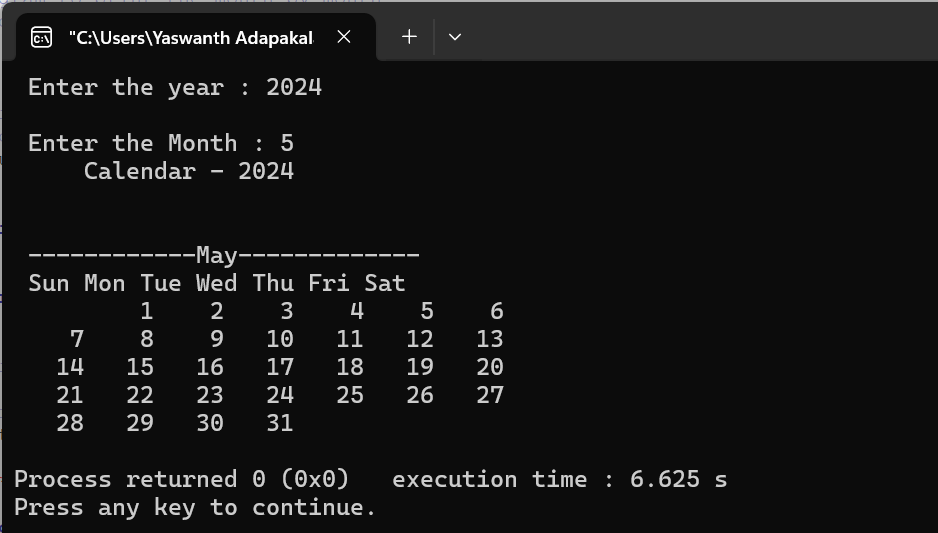
// Function Call

printCalendar(year,i);

return 0;

}

**Output :**



**Task 8 : To find the day of a given date**

**Code :**

#include <stdio.h>

int dayofweek(int d, int m, int y)

{

static int t[] = { 0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4 };

y -= m < 3;

return (y + y / 4 - y / 100 + y / 400 + t[m - 1] + d)

% 7;

}

void main()

{

int dd,mm,yyyy;

printf(" Enter the date (dd/mm/yyyy) : ");

scanf("%d/%d/%d",&dd,&mm,&yyyy);

printf("\n Day of the given date is : ");

int day = dayofweek(dd, mm, yyyy);

switch (day) {

case 0:

printf("Sunday\n");

break;

case 1:

printf("MOnday\n");

break;

case 2:

printf("Tuesday\n");

break;

case 3:

printf("Wednesday\n");

break;

case 4:

printf("Thursday\n");

break;

case 5:

printf("Friday\n");

break;

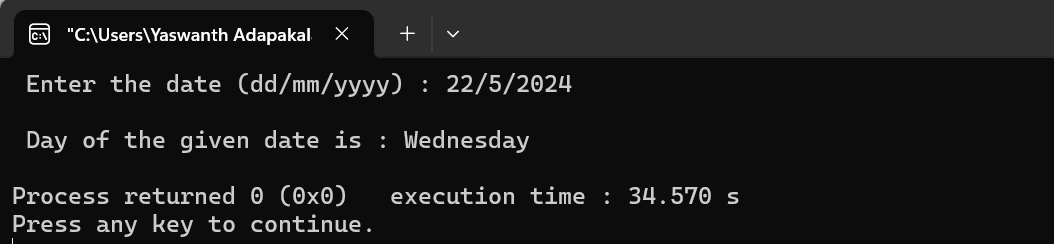
case 6:

printf("Saturday\n");

}

}

**Output :**



**Tasks on Pointers :\_**

**Task 1 :**

**Code :**

#include<stdio.h>

void main()

{

int \*p,n;

p=&n;

n=0x18;

printf(" %d %u \n",n,p);

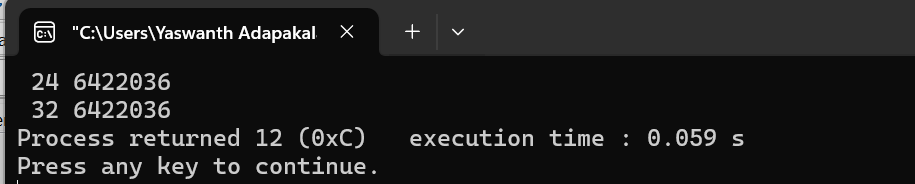
\*p=\*p+4;

n=\*p+4;

printf(" %d %u ",n,p);

}

**Output :**



**Task 2 :**

**Code :**

#include<stdio.h>

void main()

{

int x,y;

int \*ptr;

x=10;

ptr = &x;

y= \*ptr;

printf(" %d (x)is stored in : %u \n",x, &x);

printf(" %d (\*&x)is stored in : %u \n",\*&x, &x);

printf(" %d (\*ptr)is stored in : %u \n",\*ptr, ptr);

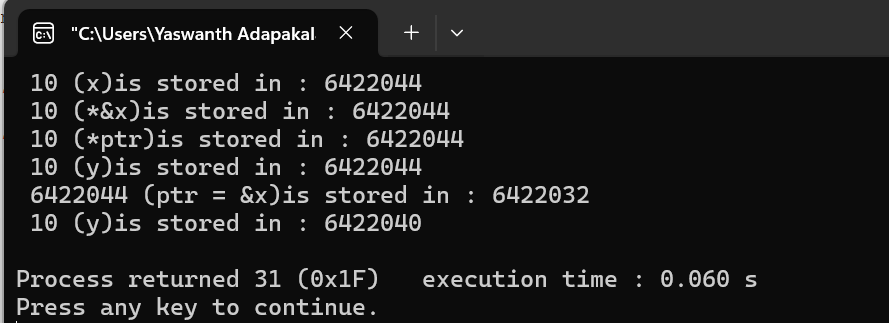
printf(" %d (y)is stored in : %u \n",y, &\*ptr);

printf(" %u (ptr = &x)is stored in : %u \n",ptr, &ptr);

printf(" %d (y)is stored in : %u \n",y, &y);

}

**Output :**



**Task 3 :**

**Code :**

#include <stdio.h>

float avg (int arr[], int size);

void main ()

{

int x[100], k, n;

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

scanf ("%d",&n);

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

for (k=0;k<n;k++)

{

scanf("%d", &x[k]);

}

printf("\n Average is : %f", avg (x,n));

}

float avg (int arr[], int size)

{

int \*p,i,sum=0;

p=arr;

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

{

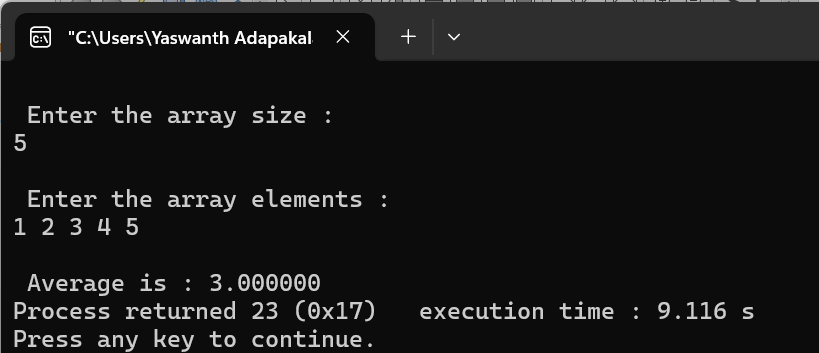
sum = sum + \*(p+i);

}

return (float) sum/size;

}

**Output :**



**Task 4 :**

**Code :**

#include <stdio.h>

#include <conio.h>

void main()

{

printf ("No. of Bytes occupied by int is %d \n", sizeof(int));

printf ("No. of Bytes occupied by float is %d \n", sizeof(float));

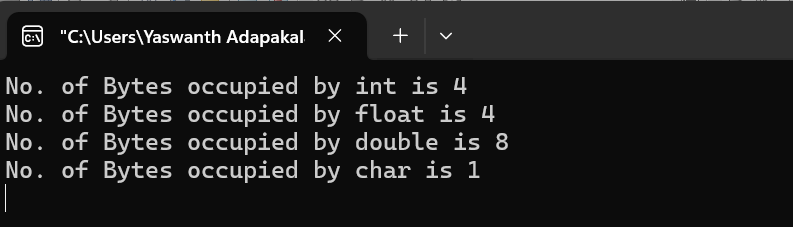
printf ("No. of Bytes occupied by double is %d \n", sizeof(double));

printf ("No. of Bytes occupied by char is %d \n", sizeof(char));

getch();

}

**Output :**



**Task 5 :**

**Code :**

#include <stdio.h>

void main()

{

int a,b;

a=5, b=20;

printf ("\n Before Swapping : \n a = %d , b = %d ", a,b);

swap (a,b);

printf ("\n Swap Fun: (call by value) \n a = %d , b = %d ",a,b);

swap1 (&a, &b);

printf ("\n Swap1 Fun: (call by Reference) \n a = %d , b = %d ",a,b);

}

void swap (int x, int y)

{

int tmp;

tmp = x;

x=y;

y=tmp;

}

void swap1 (int \*x1, int \*y1)

{

int tmp1;

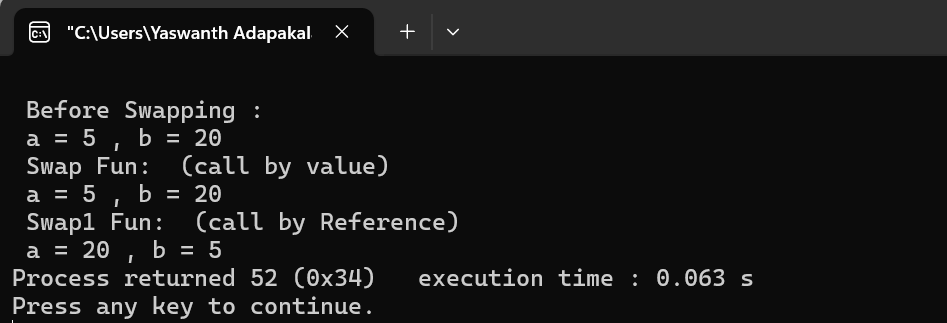
tmp1 = \*x1;

\*x1=\*y1;

\*y1=tmp1;

}

**Output :**



**Tasks on Array Sorting :\_**

**Task 1 : Bubble Sort**

**Code :**

// program for bubble sorting

#include<stdio.h>

void main()

{

int a[70],n,i,j;

printf(" Enter the size of the array : ");

scanf("%d",&n);

printf("\n Enter the %d elements of the array : ",n);

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

{

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

}

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

{

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

{

if (a[j] > a[j+1])

{

int temp = a[j];

a[j] = a[j+1];

a[j+1] = temp;

}

}

}

printf(" Array after Bubble sorting is : \n");

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

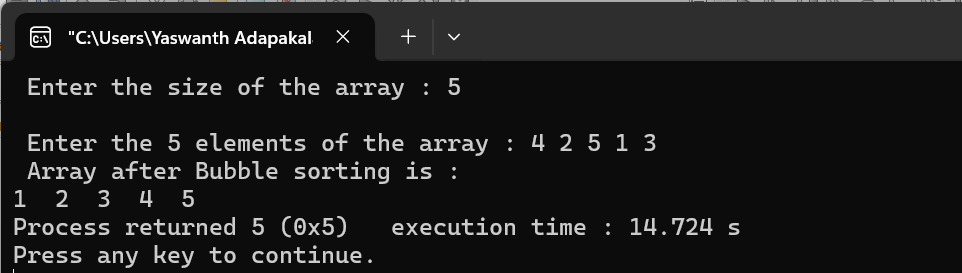
{

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

}

}

**Output :**



**Task 2 : Selection Sort**

**Code :**

// program for Selection sorting

#include<stdio.h>

void main()

{

int a[70],n,i,j;

printf(" Enter the size of the array : ");

scanf("%d",&n);

printf("\n Enter the %d elements of the array : ",n);

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

{

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

}

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

{

for (j=i+1; j < n; j++)

{

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

{

int temp = a[i];

a[i] = a[j];

a[j] = temp;

}

}

}

printf(" Array after Bubble sorting is : \n");

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

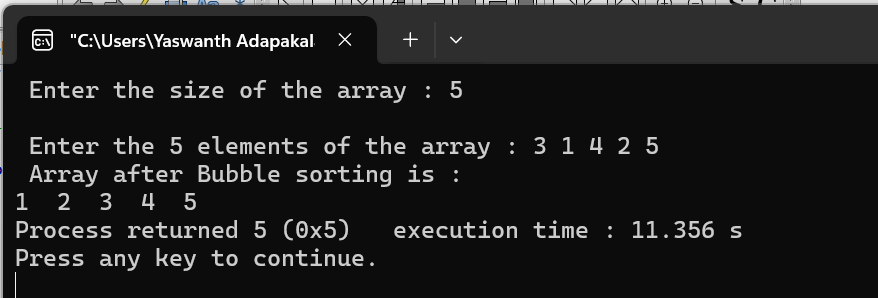
{

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

}

}

**Output :**



**Task 3 :**

**Code :**

#include <stdio.h>

void main()

{

int arr[5];

int i,j,n,counter=0, choice;

printf("Enter the Array Elements::\n");

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

{

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

}

for (j=0;j<5;j++)

{

printf("a[%d] is : : %d\n",j,arr[j]);

}

start :

printf("Enter the Array Index you want to delete::\n");

scanf("%d",&n);

arr[n] = 0;

printf("Array Elements after Deletion::\n");

for (j=0;j<5;j++)

{

printf("a[%d] is : : %d\n",j,arr[j]);

}

printf("\n Do you want to continue deletion : (0/1) : ");

scanf("%d",&choice);

if(choice==1)

{

goto start;

}

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

{

if (arr[i]== 0)

counter = counter +1;

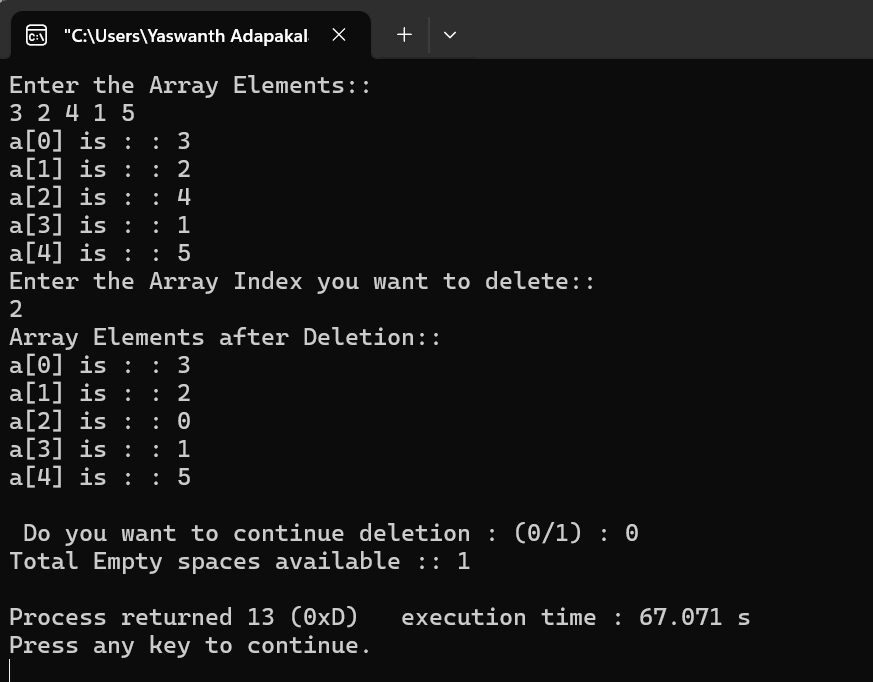
}

printf("Total Empty spaces available :: %d\n", counter);

getch();

}

**Output :**



**Task 4 :**

**Code :**

#include <stdio.h>

void main()

{

int a[5];

int i,j,n;

printf("Enter the Array Elements::\n");

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

{

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

}

printf("Enter the Element to search ::\n");

scanf("%d",&n);

for (i=0; i < 5-1; i++)

{

for (j=0; j < 5-i-1; j++)

{

if (a[j] > a[j+1])

{

int temp = a[j];

a[j] = a[j+1];

a[j+1] = temp;

}

}

}

int low =0,high=4,flag=0;;

while (low <= high)

{

int mid = low + (high - low) / 2;

if (a[mid] == n)

{

printf("\n Element is found at index position : %d \n",mid);

flag++;

}

if (a[mid] < n)

low = mid + 1;

else

high = mid - 1;

}

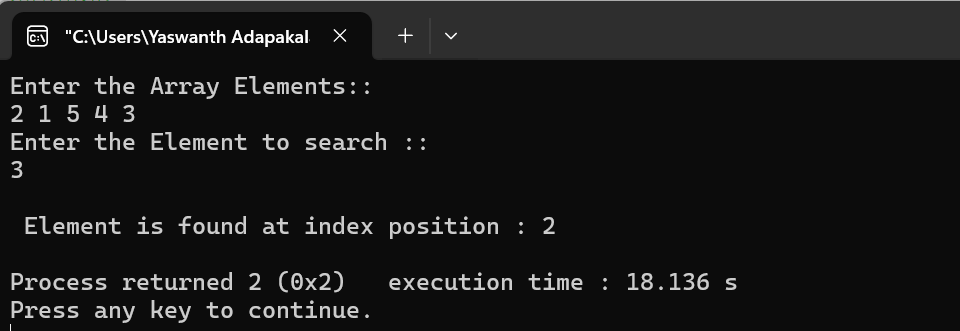
if(flag ==0){

printf(" \n Element NOT FOUND \n");

}

}

**Output :**



**Tasks on Stack and Queue :\_**

**Task 1 : Stack**

**Code :**

// Stack implementation in C

#include <stdio.h>

#include <stdlib.h>

#define MAX 10

int count = 0;

// Creating a stack

struct stack {

int items[MAX];

int top;

};

typedef struct stack st;

void createEmptyStack(st \*s) {

s->top = -1;

}

// Check if the stack is full

int isfull(st \*s) {

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

return 1;

else

return 0;

}

// Check if the stack is empty

int isempty(st \*s) {

if (s->top == -1)

return 1;

else

return 0;

}

// Add elements into stack

void push(st \*s, int newitem) {

if (isfull(s)) {

printf("STACK FULL");

} else {

s->top++;

s->items[s->top] = newitem;

}

count++;

}

// Remove element from stack

void pop(st \*s) {

if (isempty(s)) {

printf("\n STACK EMPTY \n");

} else {

printf("Item popped= %d", s->items[s->top]);

s->top--;

}

count--;

printf("\n");

}

// Print elements of stack

void printStack(st \*s) {

printf("Stack: ");

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

printf("%d ", s->items[i]);

}

printf("\n");

}

// Driver code

int main() {

int ch,n,ele;

st \*s = (st \*)malloc(sizeof(st));

createEmptyStack(s);

printf(" Enter the size of the stack : ");

scanf("%d",&n);

printf("\n Enter the elements in to the stack : ");

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

{

scanf("%d",&ele);

push(s, ele);

}

printStack(s);

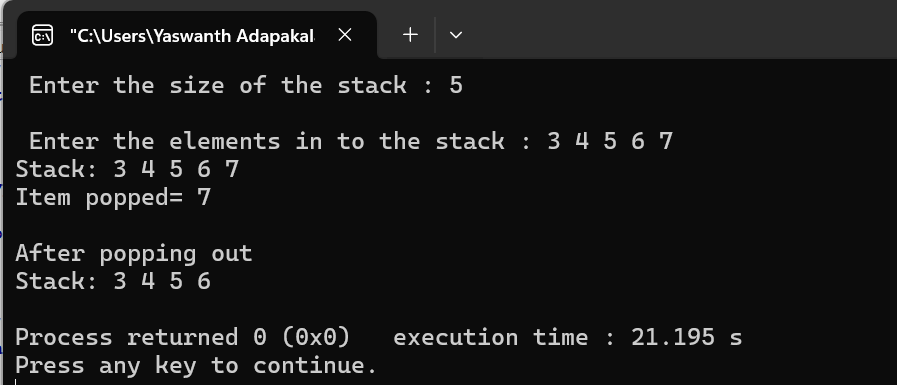
pop(s);

printf("\nAfter popping out\n");

printStack(s);

}

**Output :**



**Task 2 : Queue**

**Code :**

// Queue implementation in C

#include <stdio.h>

#define SIZE 5

void enQueue(int);

void deQueue();

void display();

int items[SIZE], front = -1, rear = -1;

int main() {

//deQueue is not possible on empty queue

deQueue();

//enQueue 5 elements

enQueue(1);

enQueue(2);

enQueue(3);

enQueue(4);

enQueue(5);

// 6th element can't be added to because the queue is full

enQueue(6);

display();

//deQueue removes element entered first i.e. 1

deQueue();

//Now we have just 4 elements

display();

return 0;

}

void enQueue(int value) {

if (rear == SIZE - 1)

printf("\nQueue is Full!!");

else {

if (front == -1)

front = 0;

rear++;

items[rear] = value;

printf("\nInserted -> %d", value);

}

}

void deQueue() {

if (front == -1)

printf("\nQueue is Empty!!");

else {

printf("\nDeleted : %d", items[front]);

front++;

if (front > rear)

front = rear = -1;

}

}

// Function to print the queue

void display() {

if (rear == -1)

printf("\nQueue is Empty!!!");

else {

int i;

printf("\nQueue elements are:\n");

for (i = front; i <= rear; i++)

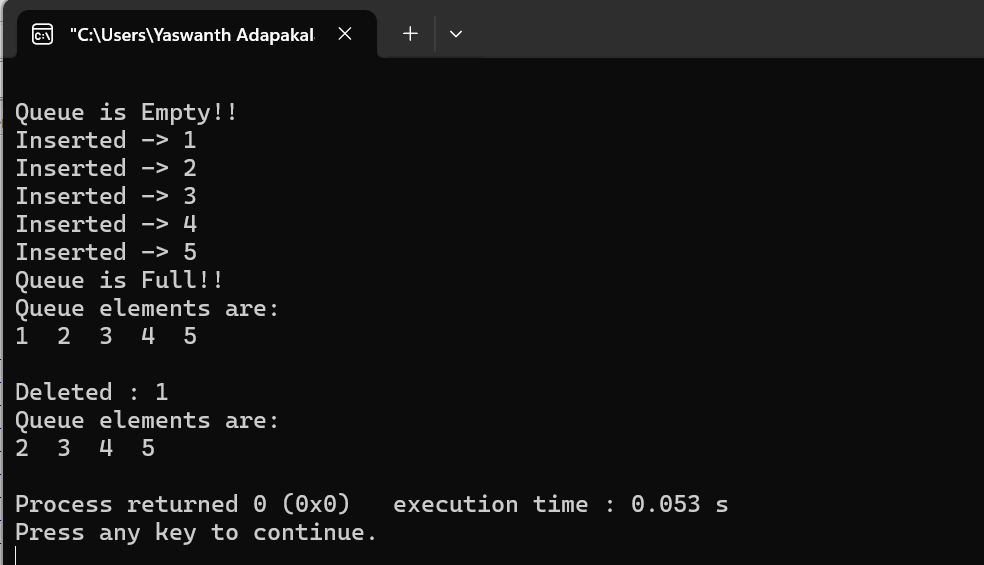
printf("%d ", items[i]);

}

printf("\n");

}

**Output :**



**Tasks on Linked lists : \_**

**Task 1 :**

**Code :**

// Linked list implementation in C

#include <stdio.h>

#include <stdlib.h>

#include<string.h>

// Creating a node

struct node {

char name[20];

int age;

struct node \*next;

};

// print the linked list value

void printLinkedlist(struct node \*p) {

while (p != NULL) {

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

printf("\n%d ",p->age);

p = p->next;

}

}

int main() {

// Initialize nodes

struct node \*head;

struct node \*one = NULL;

struct node \*two = NULL;

struct node \*three = NULL;

// Allocate memory

one = malloc(sizeof(struct node));

two = malloc(sizeof(struct node));

three = malloc(sizeof(struct node));

// Assign value values

strcpy(one->name,"one");

one->age = 1;

strcpy(two->name,"two");

two->age = 2;

strcpy(three->name,"three");

three->age = 3;

// Connect nodes

one->next = two;

two->next = three;

three->next = NULL;

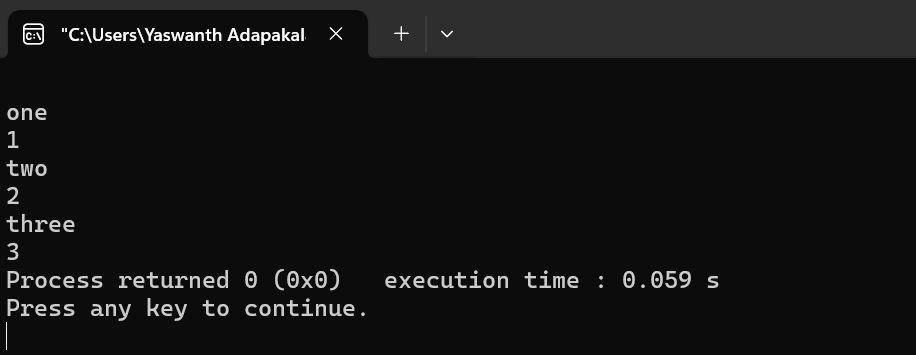
// printing node-value

head = one;

printLinkedlist(head);

}

**Output :**



**Task 2 :**

**Code :**

// Linked list implementation in C

#include <stdio.h>

#include <stdlib.h>

#include<string.h>

// Creating a node

struct node {

char name[20];

int age;

struct node \*next;

};

// print the linked list value

void printLinkedlist(struct node \*p) {

while (p != NULL) {

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

printf("\n%d ",p->age);

p = p->next;

}

}

// Assigning the linked list value

void AssignLinkedlist(struct node \*p) {

printf("\n Enter the name and age of the student : ");

while (p != NULL) {

scanf("\n%s",&p->name);

scanf("\n%d",&p->age);

p = p->next;

}

}

int main() {

// Initialize nodes

struct node \*head;

struct node \*one = NULL;

struct node \*two = NULL;

struct node \*three = NULL;

// Allocate memory

one = malloc(sizeof(struct node));

two = malloc(sizeof(struct node));

three = malloc(sizeof(struct node));

// Connect nodes

one->next = two;

two->next = three;

three->next = NULL;

head=one;

AssignLinkedlist(head);

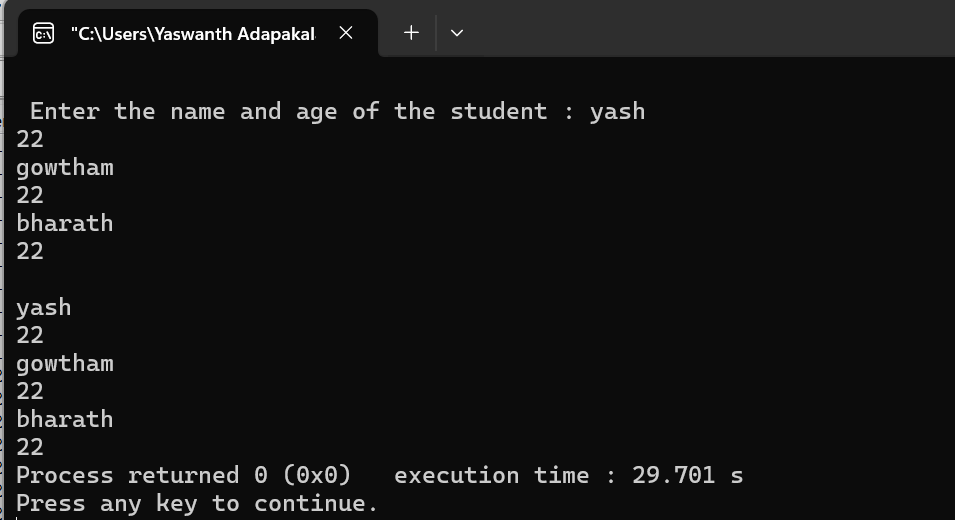
// printing node-value

head = one;

printLinkedlist(head);

}

**Output :**



**Task 3 :**

**Code :**

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* next;

};

void printLinkedlist(struct Node \*p) {

while (p != NULL) {

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

p = p->next;

}

}

void main()

{

struct Node\* head = NULL;

int count;

printf(" Enter the no. of lists you want to enter : ");

scanf("%d",&count);

for(int i=0;i<count;i++)

{

struct Node \*newNode;

newNode = malloc(sizeof(struct Node));

printf("\n Enter the data : ");

int num;

scanf("%d",&num);

newNode->data = num;

newNode->next = head;

head = newNode;

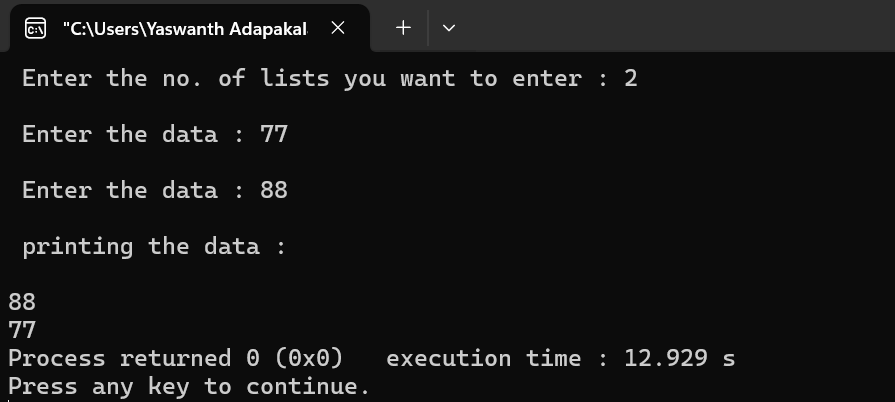
}

printf("\n printing the data : \n");

printLinkedlist(head);

}

**Output :**



**Binary Tree : \_**

**Code :**

// binary tree

#include <stdio.h>

#include <stdlib.h>

struct Node {

int a;

struct Node \*left;

struct Node \*right;

};

struct Node \*root = NULL;

struct Node\* insert() {

int data;

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

printf("Enter Data [-1 for start inserting left or right]\n");

scanf("%d", &data);

if(data == -1)

return 0;

nn->a = data;

printf("Enter Left Node Data ");

nn->left=insert();

printf("Enter Right Node Data ");

nn->right=insert();

return nn;

}

void preorder(struct Node \*root) {

if (root == NULL) {

return;

}

printf("%d ", root->a);

preorder(root->left);

preorder(root->right);

}

void inorder(struct Node \*root) {

if(root == NULL) {

return ;

}

inorder(root->left);

printf("%d ", root->a);

inorder(root->right);

}

void postorder(struct Node \*root) {

if(root == NULL) {

return ;

}

postorder(root->left);

postorder(root->right);

printf("%d ", root->a);

}

int main() {

root = insert();

printf("Printing the data in the list[preOrder Traversal]\n");

preorder(root);

printf("\nPrinting the data in the list[inOrder Traversal]\n");

inorder(root);

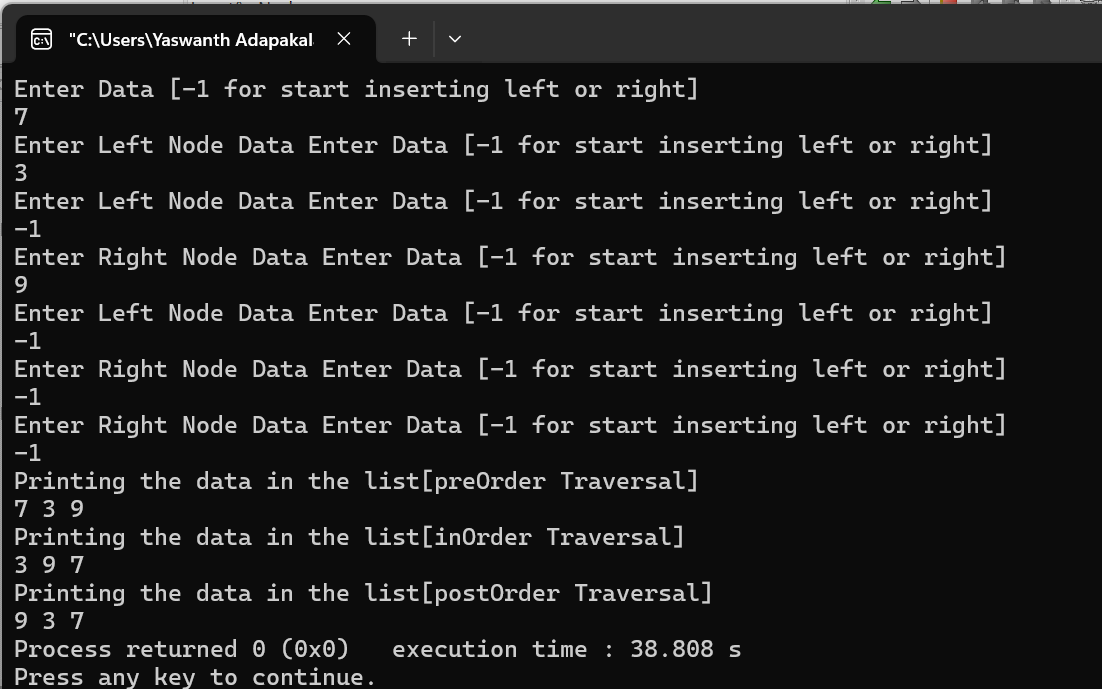
printf("\nPrinting the data in the list[postOrder Traversal]\n");

postorder(root);

return 0;

}

**Output :**

****

**Tasks on Structures : \_**

**Task 1 :**

**Code :**

#include <stdio.h>

struct person{

int age;

float weight;

};

int main(){

struct person \*personPtr, person1;

personPtr = &person1;

printf("Enter age: ");

scanf("%d", &personPtr->age);

printf("Enter weight: ");

scanf("%f", &personPtr->weight);

printf("Displaying:\n");

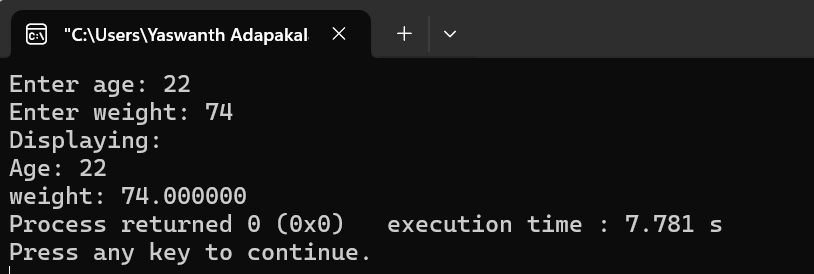
printf("Age: %d\n", personPtr->age);

printf("weight: %f", personPtr->weight);

return 0;

}

**Output :**



**Task 2 :**

**Code :**

#include <stdio.h>

#include <stdlib.h>

struct person {

int age;

float weight;

char name[30];

};

int main()

{

struct person \*ptr;

int i, n;

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

scanf("%d", &n);

// allocating memory for n numbers of struct person

ptr = (struct person\*) malloc(n \* sizeof(struct person));

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

{

printf("Enter first name and age respectively: ");

// To access members of 1st struct person,

// ptr->name and ptr->age is used

// To access members of 2nd struct person,

// (ptr+1)->name and (ptr+1)->age is used

scanf("%s %d", (ptr+i)->name, &(ptr+i)->age);

}

printf("Displaying Information:\n");

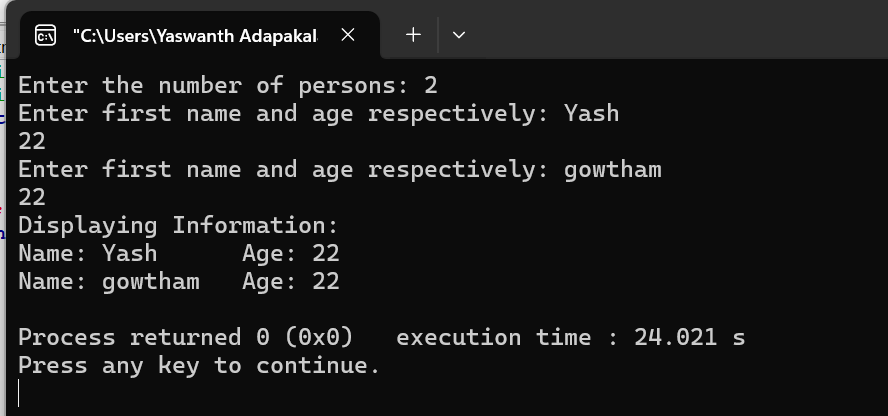
for(i = 0; i < n; ++i)

printf("Name: %s\tAge: %d\n", (ptr+i)->name, (ptr+i)->age);

return 0;

}

**Output :**



**Task 3 :**

**Code :**

#include <stdio.h>

struct student {

char name[50];

int age;

};

// function prototype

void display(struct student s);

int main() {

struct student s1;

printf("Enter name: ");

// read string input from the user until \n is entered

// \n is discarded

scanf("%[^\n]%\*c", s1.name);

printf("Enter age: ");

scanf("%d", &s1.age);

display(s1); // passing struct as an argument

return 0;

}

void display(struct student s) {

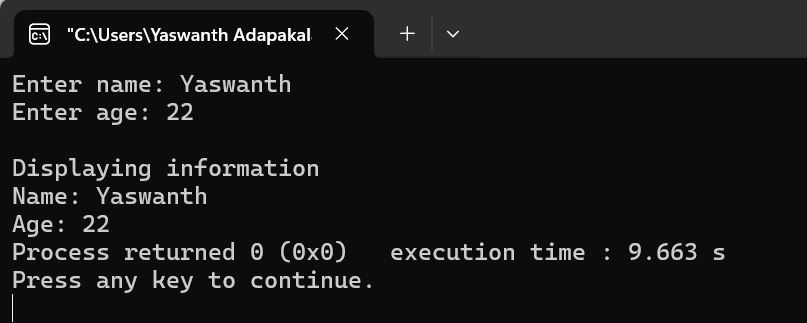
printf("\nDisplaying information\n");

printf("Name: %s", s.name);

printf("\nAge: %d", s.age);

}

**Output :**



**Task 4 :**

**Code :**

#include <stdio.h>

struct name {

int member1;

int member2;

};

int main()

{

struct name \*ptr, Harry;

printf("Size of struct name: %zu bytes\n", sizeof(struct name));

printf("Size of member1: %zu bytes\n", sizeof(Harry.member1));

printf("Size of member2: %zu bytes\n", sizeof(Harry.member2));

ptr=&Harry;

printf("addres of name %zu",ptr);

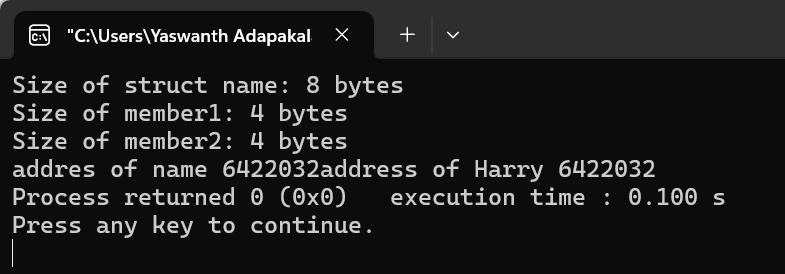
ptr=&Harry.member1;

printf("address of Harry %zu",ptr);

return 0;

}

**Output :**



**Tasks using Recursion : \_**

**Task 1 : Fibonacci series using Recursion**

**Code :**

#include <stdio.h>

int fibonacci(int n) {

if(n == 0)

return 0;

else if(n == 1)

return 1;

else

return (fibonacci(n-1) + fibonacci(n-2));

}

int main() {

int n;

printf("Enter the number of terms\n");

scanf("%d", &n);

printf("Fibonacci Series: ");

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

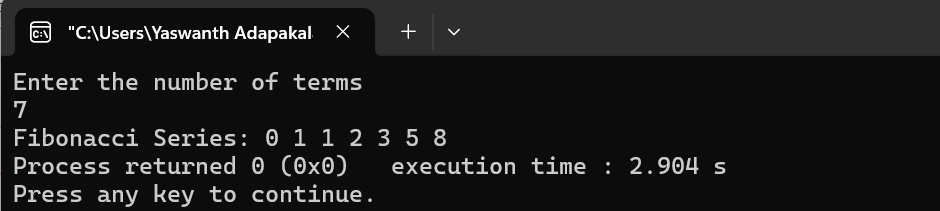
printf("%d ", fibonacci(i));

}

return 0;

}

**Output :**



**Task 2 : Sum of n numbers using recursion**

**Code :**

#include <stdio.h>

int sum(int n);

int main() {

int number, result;

printf("Enter a positive integer: ");

scanf("%d", &number);

result = sum(number);

printf("sum = %d", result);

return 0;

}

int sum(int n) {

if (n != 1)

// sum() function calls itself

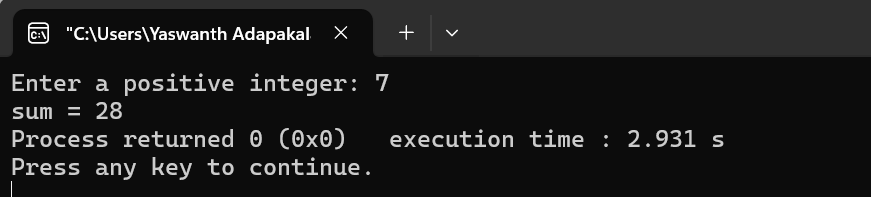
return n + sum(n-1);

else

return n;

}

**Output :**



**Task 3 : Factorial using recursion**

**Code :**

#include <stdio.h>

int sum(int n);

int main() {

int number, result;

printf("Enter a positive integer: ");

scanf("%d", &number);

result = sum(number);

printf("\n Factorial of given number is : %d", result);

return 0;

}

int sum (int n) {

if (n != 1)

// sum() function calls itself

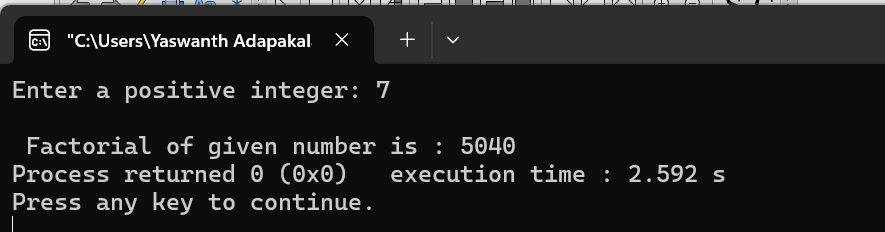
return n \* sum(n-1);

else

return n;

}

**Output :**



**Tower of Hanoi : \_**

**Task : Solving tower of Hanoi using recursion**

**Code :**

#include <stdio.h>

void hanoi (int n, char from, char to, char via) {

if (n == 1) {

printf ("Move disk 1 from %c to %c\n", from, to);

}

else {

hanoi(n-1, from, via, to);

printf("Move disk %d from %c to %c\n", n, from, to);

hanoi (n-1, via, to, from);

}

}

int main () {

int n = 3;

char from = 'A';

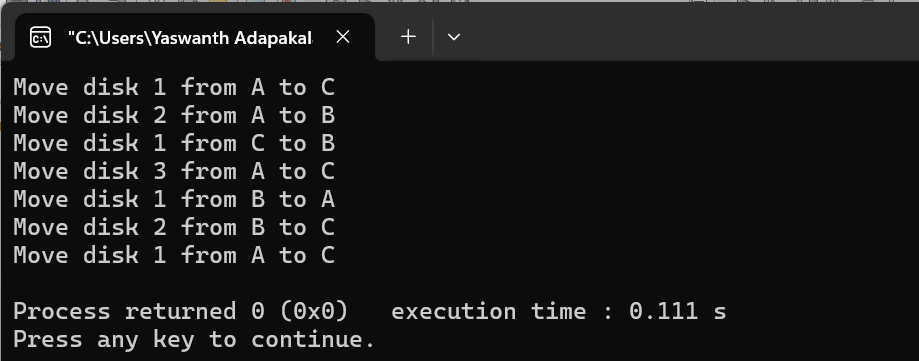
char to = 'B';

char via = 'C';

hanoi (n, from, via, to); //calling hanoi() method

}

**Output :**



**\_\_\_\_\_\_\_\_\_\_THANK YOU\_\_\_\_\_\_\_\_\_\_**