**SCHOOL OF ENGINEERING AND TECHNOLOGY**

**SRI PADMAVATI MAHILA VISVAVIDYALAYAM**

**Tirupati**

***Lab Record***

**Department of Computer Science and Engineering**

****

Name of the Student:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Roll Number:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Year:\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Semester:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Lab Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**SRI PADMAVATI MAHILA VISVAVIDYALAYAM,TIRUPATI**

**(Women’s University)**

**SCHOOL OF ENGINEERING & TECHNOLOGY**

****

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**Certificate**

**Roll Number:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Certified that is the bonafide work of**

**Mrs/Miss\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Of the year\_\_\_\_\_\_\_\_\_\_\_sem\_\_\_\_\_\_\_\_\_\_\_\_\_Branch\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**And has conducted \_\_\_\_\_\_\_\_\_\_\_\_\_\_out of\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ experiments in\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Laboratory and has undergone the course of Practical instructions during the Academic year\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Staff – In- charge Head of the Department**

**Date: Examiner**

**SCHOOL OF ENGINEERING AND TECHNOLOGY,SPMVV,TIRUPATI**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| INDEX | | | | | |
| **S.No.** | **Date** | **Name Of The Experiment** | **Page No.** | **Marks** | **Remarks** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**SCHOOL OF ENGINEERING AND TECHNOLOGY,SPMVV,TIRUPATI**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| INDEX | | | | | |
| **S.No.** | **Date** | **Name Of The Experiment** | **Page No.** | **Marks** | **Remarks** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**SCHOOL OF ENGINEERING AND TECHNOLOGY,SPMVV,TIRUPATI**

## ARITHMETIC OPERATIONS

Experiment No.1 Roll no. 1872062

Date: Page no.

Aim: To find Arithmetic operations in C programming

Algorithm:

Step 1:Start.

Step 2: Declare a, b, add, sub, mul,div,mod as integers

Step 3: Read a, b values

Step 4: add⇓a+b, sub⇓a-b, mul⇓a\*b, div⇓a/b, mod⇓a%b

Step 5:print result

Step 6: Stop

Source code:

#include<stdio.h>

#include<conio.h>

void main()

{

int add,sub,mul,mod;

float div;

clrscr();

printf("enter a and b values");

scanf("%d%d",&a,&b);

add=a+b;

sub=a-b;

mul=a\*b;

div=(float)a/b;

mod=a%b;

printf("\naddition oftwo numbers%d",add);

printf("\ndifference of two nubers%d",sub);

printf("\nproduct of two numbes%d",mul);

printf("\ndivision of two numbers%f",div);

printf("\nmodulus division of two numbers%d",mod)

getch();

}

Expected output:

Enter a and b values: 33 4

Addition of two numbers 37

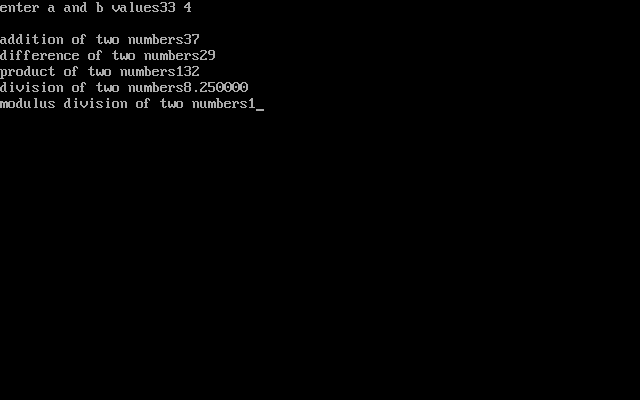
Difference of two numbers 29

Product of two numbers 132

Division of two numbers 8.25

Modulus division of two numbers 1

Actual ouput:



Result: Above program has been executed successfully.

## SIMPLE INTEREST

Experiment No.2 Roll no.1872062

Date: Page no.

Aim: To find the simple interest in C programming

Algorithm:

Step 1: Start.

Step 2: Declare p, t, r as integers and n as floating value

Step 3: Read p, t, r

Step 4: n⇓(p\*t\*r) /100

Step 5:print result

Step 6: Stop

Source code:

#include<stdio.h>

#include<conio.h>

void main()

{

int p,t,r;

float s;

clrscr();

printf("\nenter p value");

printf("\nenter t value");

printf("\nenter r value");

scanf("%d",&p);

scanf("%d",&t);

scanf("%d",&r);

s=(p\*t\*r)/100;

printf("\nsimple interestis:%f",s);

getch();

}

Expected Output:

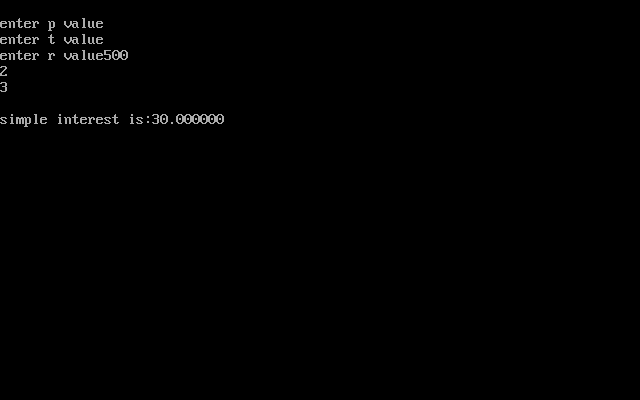
Enter p value 500

Enter t value 2

Enter r value 3

Simple interest:30

Actual Output:



Result: Above program has been executed successfully.

## COMPOUND INTEREST

Experiment No.3 Roll no.1872062

Date: Page no.

Aim: To find the Compound interest in C programming

Algorithm:

Step 1: Start.

Step 2: Declare p, t, r, n as floating values

Step 3: Read p, t, r

Step 4: n⇓p\*(pow( ( 1+r/100), t))

Step 5:print result

Step 6: Stop

Source code:

#include<stdio.h>

#include<conio.h>

#include<math.h>

void main()

{

float p,r,n,c;

clrscr();

printf("enter p,r,n,c");

scanf("%f%f%f%f",&p,&r,&n,&c);

c=p\*(pow((1+r/100),n));

printf("compound interest %f",c);

getch();

}

Expected input:

Enter p,r,n 2 4 5

Compound interest is 2.433306

Actual output:



Result: Above program has been executed successfully.

## AREA OF CIRCLE

Experiment No.4 Rollno.1872062

Date: Page no.

Aim: To find the area of circle in C programming

Algorithm:

Step 1: Start.

Step 2: Declare r, pi=3.14, n as float values

Step 3: Read r

Step 4:n⇓pi\*r\*r

Step 5:print area of the circle

Step 6: Stop

Source Code:

#include<stdio.h>

#include<conio.h>

#include <math.h>

void main()

{

float r,pi=3.14,a;

clrscr();

printf("enter r value");

scanf("%d",&r);

a=(pi\*r\*r);

printf("area of the circle %f",a);

getch();

}

Expected output:

Enter r value: 12

Area of circle 452.160004

Actual output:



Result: Above program has been executed successfully.

## To read in two integers and display one as percentage of the other.

Experiment No.5 Rollno.1872062

Date: Page no.

Aim: To write a c program to read in two integers and display one as a percentage of the other. Typically, your output should look like 20 is 50.00% of 40 assuming that the input numbers were 20 and 40 display the percentage correct to 2 decimals.

Algorithm:

Step 1: Start.

Step 2: Declare two integers and one float variable to store the output.

Step 3: check the condition if a<b or a>b and find out percentage a/b\*100 and convert it into float value.

Step 4: store the value and print it.

Step 5: stop

Source code:

#include<stdio.h>

#include<conio.h>

void main()

{

int a,b;

float c,d,e;

clrscr();

printf("enter a and b values:");

scanf("%d %d",&a,&b);

if(a<b)

{

c=(float)a/b\*100;

printf("%d is %.2f%% of %d",a,c,b);

}

else

{

c=(float)b/a\*100;

printf("%d is %.2f of %d",b,c,a);

}

getch();

}

Expected output:

Enter a and b values: 75 50

50 is 66.67 of 75

Actual output:



Result: Above program has been executed successfully.

## To output the possible roots of a given set of coefficients

Experiment No.6 Rollno.1872062

Date: Page No

Aim: To write a c program to output the possible roots of a given set of coefficients with appropriate messages.

Algorithm:

Step 1: Start.

Step 2: Read a, b, c as integer.

Step 3: declare root1, root2 as float.

Step 4: if(b\*b>4\*a\*c)

Print “the roots are real roots”

Assign root1 ⇓((-b/2\*a) + sqrt(b\*b-4\*a\*c)/2\*a)

Assign root2⇓((-b/2\*a) – sqrt(b\*b-4\*a\*c)/2\*a)

Print root1, root2

else if(b\*b=4\*a\*c)

print “the roots are repeated roots”

assign root1⇓-b/2\*a print root1

else

print “the roots are imaginary”

Step 5: Stop.

Source code:

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#include<math.h>

void main()

{

float a, b, c, root1, root2;

float realp, imagp;

clrscr();

printf("Enter the values of a, b and c \n");

scanf("%f %f %f", &a, &b, &c);

if (a == 0 || b == 0 || c == 0)

{

printf("Error: Roots cannot be determined \n");

exit(1);

}

else

{

if (b\*b-4.0\*a\*c< 0)

{

printf("Imaginary Roots\n");

realp = -b / (2.0 \* a) ;

imagp = sqrt(abs(b\*b-4.0\*a\*c)) / (2.0 \* a);

printf("Root1 = %f +i %f\n", realp, imagp);

printf("Root2 = %f -i %f\n", realp, imagp);

}

else if (b\*b-4.0\*a\*c== 0)

{

printf("Roots are real and equal\n");

root1 = -b / (2.0 \* a);

root2 = root1;

printf("Root1 = %f\n", root1);

printf("Root2 = %f\n", root2);

}

else if (b\*b-4.0\*a\*c > 0 )

{

printf("Roots are real and distinct \n");

root1 =(-b + sqrt(b\*b-4.0\*a\*c)) / (2.0 \* a);

root2 =(-b - sqrt(b\*b-4.0\*a\*c)) / (2.0 \* a);

printf("Root1 = %f \n", root1);

printf("Root2 = %f \n", root2);

}

}

getch();

}

Expected output:

Enter a,b and c values 20 -15 -10

Roots are real and distinct

Root1=1.175390

Root2=-0.425391

Actual output:



Result: Above program has been executed successfully.

## Date -Month-Year

Experiment No.7 Rollno.1872062

Date: Page No.

Aim: To write a c program taken input of date as day, month, year and print the output of the following date.

Algorithm:

Step 1: Start.

Step 2: Declare variables, day, month, year.

Step 3: enter date.

Step 4: check condition for even month or odd month.

Step 5: if it is odd month check condition

If(day==1&&day<=30) then day=day+1, else

Month=month+1&day=1.

Step 6: if it is even month but not equal to 2 then check condition

if(day<=1&&day<=29) then day=day+1,else

month=month+1 & day=1.

Step 7: if month==2 then check for leap year, if it is leap year then check condition

If (day==1&&day<=28) then day=day+1 else month=month+1 & day=1

else check condition if(day==1&&day<=27) then day=day+1

Step 8: print date

Step 9: Stop.

Source code:

#include<stdio.h>

#include<conio.h>

void main()

{

int mon,day,year;

clrscr();

printf("enter date");

scanf("%d%d%d",&day,&mon,&year);

if(mon==1||mon==3||mon==5||mon==7||mon==8||mon==10);

{

if((day==1)||(day<=30))

{

day=day+1;

}

else

{

day=1;

mon=mon+1;

}

}

if(mon==4||mon==6||mon==9||mon==11)

{

if((day==1)||(day<=29))

{

day=day+1;

}

else

{

day=1;

mon=mon+1;

}

}

if(mon==2)

{

if((year%4)==0)

{

if((day==1)||(day<=28))

{

day=day+1;

}

else

{

day=1;

mon=mon+1;

}

}

else if((day==1)||(day<=28))

{

day=day+1;

}

else

{

day=1;

mon=mon+1;

}

}

if(mon==12)

{

if((day==1)||(day<=30))

{

day=day+1;

}

else

{

mon=1;

day=1;

year=year+1;

}

}

printf("the following date is %d:%d:%d",day,mon,year);

getch();

}

Expected output:

Enter date: 01 08 2001

The following date is 02:08:2001

Actual output:



Result: Above program has been executed succeesfully.

## Amicable or not

Experiment No.8 Roll no. 1872062

Date: Page No.

Aim: To write a c program to check whether given two numbers are amicable or not.

Algorithm:

Step 1: Start.

Step 2: declare num1, num2.

Step 3: initialize num1divisorsum=0 and num2divisorsum=0

Step 4: for(int i=1;i<num1;i++)

{

If(num1%i==0)

{

Num1divisorsum=num1divisorsum+i;

}

}

Step 5: for(int i=1;i<num2;i++)

{

If(num2%i==0)

{

Num2divisorsum=num2divisorsum+i;

}

}

Step 6: if((num1==num2divisorsum)&&(num2==num1divisorsum)) is true go to step 7,else

Go to step 8.

Step 7: print two numbers are amicable

Step 8: print two numbers are not amicable.

Step 9:stop

Source code:

#include<stdio.h>

#include<conio.h>

void main()

{

int i,num1,num2,num1divisorsum=0, num2divisorsum=0;

clrscr();

printf("enter two numbers to check if amicable or not :");

scanf("%d %d",&num1,&num2);

for(i=1;ii<num1;i++)

{

if(num1%i==0)

{

num1divisorsum=num1divisorsum+i;

}

}

for(i=1;i<num2;i++)

{

if(num2%i==0)

{

num2divisorsum=num2divisorsum+i;

}

}

if((num1==num2divisorsum)&&(num2=num1divisorsum))

{

printf(“%d and %d are amicable numbers\n”,num1,num2);

}

else

{

printf(“%d and %d are not amicable numbers\n”,num1,num2);

}

getch();

}

Expected output:

enter two numbers to check if amicable or not : 60 80

60 and 80 are not amicable numbers

Actual output:



Result: Above program has been executed successfully.

## STUDENT MARK GRADING

Experiment No.9 Roll no.1872062

Date: Page no.

Aim: To write a c program for students mark grading.

Algorithm:

Step 1: Start.

Step 2: declare a,b,c,d,tot,avg.

Step 3: enter marks in all subjects.

Step 4: if (a>34&&b>34&&c>34&&d>34) is true then print student gets passed else go to

Step 5.

Step 5: print student is failed.

Step 6: calculate tot = a + b + c + d and avg =(float)tot/4.

Step 7: if(avg>=90) is true then print student got grade A else go to step 8.

Step 8: else if(avg>=80&&avg=90) then Print student got grade: A+.

Step 9: else if(avg>=70&&avg=80) then Print student got grade: B

Step 10: else if(avg>=60&&avg=70) then Print student got grade: B+

Step 11: else if(avg>=50&&avg=60) then Print student got grade: C

Step 12: else if(avg>=40&&avg=50) then Print student got grade: D

Step 13: else if(avg>=35&&avg=40) then Print student got grade: E

Step 14: else Print student got grade F.

Step 15: Print the average of a student.

Step 16: stop.

Source code:

#include<stdio.h>

#include<conio.h>

void main()

{

int a,b,c,d,tot;

float avg;

clrscr();

printf("enter marks in all subjects");

scanf("%d %d %d %d",&a,&b,&c,&d);

if(a>34&&b>34&&c>34&&d>34)

{

printf("student gets passed\n");

}

else

{

printf("student get failed\n");

}

tot=a+b+c+d;

avg=(float)tot/4;

if(avg>=90)

{

printf("student got grade:'A'");

}

else if(avg>=80&&avg<90)

{

printf("student got grade:'A+'");

}

else if(avg>=70&&avg<80)

{

printf("student got grade:'B'");

}

else if(avg>=60&&avg<70)

{

printf("student got grade:'B+'");

}

else if(avg>=50&&avg<60)

{

printf("student got grade:'C'");

}

else if(avg>=40&&avg<50)

{

printf("student got grade:'D'");

}

else if(avg>=35&&avg<40)

{

printf("student got grade:'E'");

}

else

{

printf("student got grade:'F'");

}

printf("\nthe average of a student is %f",avg);

getch();

}

Expected output:

enter marks in all subjects 90

80

75

60

Student get passed

Student got grade B

The average of student is 76.25

Actual output:



Result: Above program has been executed successfully.

## Palindrome or not

Experiment No.10 Roll no.1872062

Date: Page no.

Aim: To write a c program that takes an integer number as input, check whether it is palindrome or not and output the reverse of the same with suitable messages.

Algorithm:

Step 1: Start.

Step 2: declare n, reversed integer, remainder and original integer.

Step 3: check the condition while(n!=0) then calculate

remainder=n%10,

reversed integer =reversed integer\*10+remainder,

n=n/10.

Step 4: if (original integer==reversed integer)

Print integer is a palindrome else go to step5

Step 5: print integer is not a palindrome.

Step 6: Stop.

Source code:

#include<stdio.h>

#include<conio.h>

void main()

{

int n,reversedInteger=0,remainder,originalInteger;

clrscr();

printf("enter an integer:");

scanf("%d",&n);

originalInteger=n;

while(n!=0)

{

remainder=n%10;

reversedInteger=reversedInteger\*10+remainder;

n=n/10;

}

if(originalInteger==reversedInteger)

printf("Num: %d,Reverse: %d,is a palindrome",originalInteger,reversedInteger);

else

printf("Num: %d,Reverse: %d,not a palindrome",originalInteger,reversedInteger);

getch();

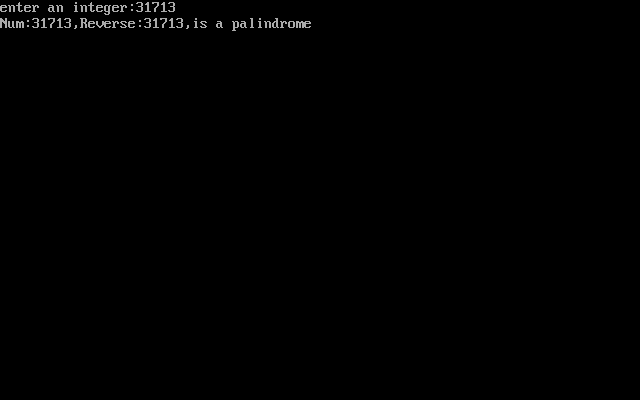
}

Expected output:

enter an integer 31713

Num: 31713 Reverse: 31713, is a palindrome.

Actual output:



Result: Above program has been executed successfully.

## Income tax

Experiment No: 11 Roll no. 1872062

Date: Page no.

Aim: To write a c program for the problem, Assume that te united states of America uses the following income tax code formula for their annual income: First US $5000 of income: 0% tax Next US $10,000 0f income: 10% of tax, next US $20,000 of income:15% tax. An amount of above US $35,000: 20% of tax. For example, somebody earning US $38,000 annually would owe US $5000\*0.00+10,000\*0.10+20,000\*0.15+30,000\*0.20, which comes to US $4600.Write a program that uses a loop to input the income and calculate and repeat the owed tax amount.

Algorithm:

Step 1: Start.

Step 2: declare float income tax=0

Step 3: enter your income.

Step 4: check the condition if(income>35000) then calculate

income=income-35000,

tax=(10,000\*0.10)+(20,000\*0.15)+(income\*0.20);

print your tax is: else go to step 5.

Step 5: else if(income>20000 && income<=35000)

income=income-15000

tax=(10,000\*0.10)+(income\*0.15) and print your tax is: else go to step 6

Step 6: else if(income>10000 && income<=20000)

income=income-10000

tax=(1000\*0.10)+(income\*0.15) and print your tax is: else go to step7

Step 7: else if(income>5000 && income<=10000)

income=income-5000

tax=income\*0.10,print your tax is: else go to step 8

Step 8: print you have no tax

Step 9: stop.

Source code:

#include<stdio.h>

#include<conio.h>

void main()

{

float income,tax=0;

clrscr();

printf("\nenter your income:US$ ");

scanf("%f",&income);

if(income>35000)

{

income=income-35000;

tax=(10000\*0.10)+(20000\*0.15)+(income\*0.20);

print("\nyour tax is US$ %f",tax);

}

else if(income>20000 && income<=35000)

{

income=income-15000;

tax=(10000\*0.10)+(income\*0.15);

printf("\nyour tax is US$ %f",tax);

}

else if(income>10000 && income<=20000)

{

income=income-15000;

tax=(10000\*0.10)+(income\*0.15);

printf("\your tax is US$ %f",tax);

}

else if(income>5000 && income< =10000)

{

income=income-5000;

tax=income\*0.10;

printf("\n your tax is US$ %f",tax);

}

else

{

printf("\n you have no tax");

}

getch();

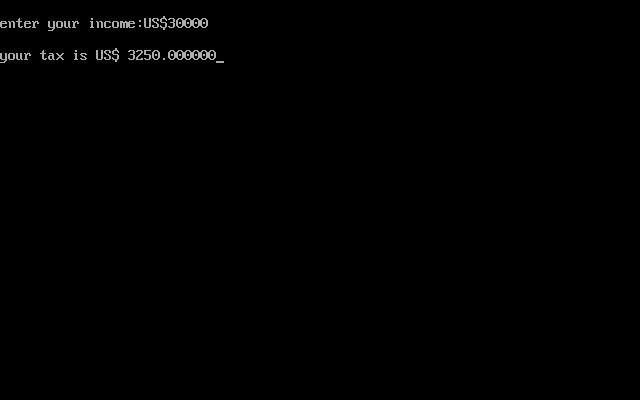
}

Expected output:

Enter your income US $30000

Your tax is US $3250

Actual output:



Result: Above program has been executed successfully.

## Hailstone sequence

Experiment No.12 Roll no.1872062

Date: Page no.

Aim: To write a c program to generate hailstone sequence.

Algorithm:

Step 1: Start.

Step 2: declare a function to generate next number.

Step 3: declare I = num.

Step 4: if (i%2 == 0) is true then calculate i=i/2 else go to step 5

Step 5: calculate i=3\*i+1.

Step 6: Stop.

Source code:

#include<stdio.h>

#include<conio.h>

int getnextnumber(int num)

{

int i=num;

if(i%2==0)

{

i=i/2;

}

else

{

i=3\*i+1;

}

return(i);

}

void getHailstone(int num)

{

int a=num;

if(a==1)

{

printf("\t%i",a);

}

else

{

printf("\t%i",a);

getHailstone(getnextnumber(a));

}

}

int main()

{

int num;

clrscr();

printf("the following sequence of numbers will be generated for n= ");

scanf("\t%i",&num);

getHailstone(num);

getch();

return 0;

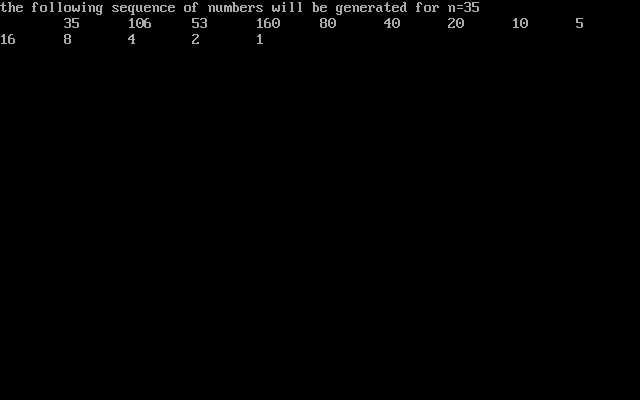
}

Expected output:

The following sequence of numbers will be generated for n= 35

35 106 53 160 80 40 20 10 5 16 8 4 2 1

Actual output:



Result: Above program has been executed successfully.

## Perfect number

Experiment No.13 Roll no. 1872062

Date: Page no.

Aim: To write a c program to check whether a given number is perfect number or not.

Algorithm:

Step 1: start.

Step 2: declare number,rem,sum=0,i.

Step 3: enter a number.

Step 4: for(i=1;i<=(number-1);i++)

rem=number%i;

If(rem==0) is true go to step 5.

Step 5: sum=sum+i.

Step 6: if(sum==number) is true go to step 7 else go to step 8.

Step 7: print entered number is a perfect number.

Step 8: print entered number is not a perfect number.

Step 9: stop.

Source code:

#include<stdio.h>

#include<conio.h>

int main()

{

int number,rem,sum=0,i;

clrscr();

printf("enter a number\n");

scanf("%d",&number);

for(i=1;i<=(number-1);i++)

{

rem=number%i;

if(rem==0)

{

sum=sum+i;

}

}

if(sum==number)

{

printf("entered number is perfect number");

}

else

{

printf("entered number is not a perfect number");

}

getch();

return 0;

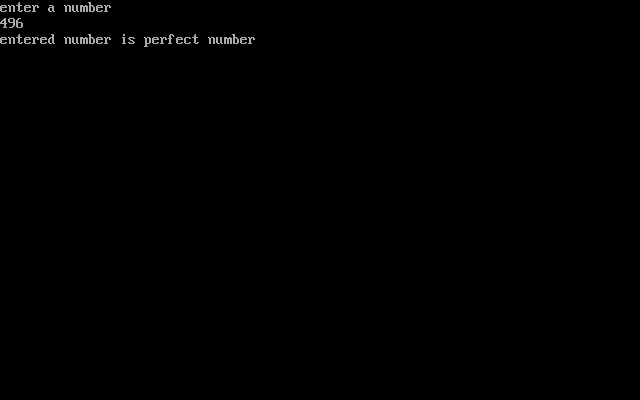
}

Expected output:

enter a number 496

entered number is a perfect number.

Actual output:



Result: Above program has been executed successfully.

## Prime numbers in the give range

Experiment No.14 Roll no.1872068

Date: Page no.

Aim: To write a c program to check whether a given number is prime number in the given

range.

Algorithm:

Step1:start

Step2:Initialize i, j, n, count=0 as integers

Step3:Read n value

Step4:for(i=1;i<=n;i++)

count=0

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

ff(i%j==0)

count++

if(count==2)

Step 5:print result

Step 6:Stop.

Source code:

#include<stdio.h>

#include<conio.h>

void main()

{

int i,j,n,count=0;

printf("enter range of prime numbers");

scanf("%d",&n);

printf("prime numbers are:");

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

{

count=0;

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

{

if(i%j==0)

{

count++;

}

}

if(count==2)

{

printf("%d",i);

}

}

getch();

}

Expected output:

enter range of prime numbers 25

prime numbers are 2 3 5 7 11 13 17 19 23

Actual output:



Result: Above program has been executed successfully.

## Call by value

Experiment No.15 Roll No.1872062

Date: Page no.

Aim: To determine the swapping of two numbers using call by value

Algorithm:

Step 1: start

Step 2: initializing a and b values

Step 3: printing the a ,b values using swapping

Step 4: calling the function swap by passing the a, b values

Step 5: swapping the a, b values using temp

Step 6: printing the a, b values after swapping the values in the function

Step 7: printing the a, b values after swapping the values in the main

Step 8: stop

Source code:

#include<stdio.h>

#include<conioh>

void swap(int , int);

void main()

{

int a = 25;

int b = 40;

printf(“Before swapping the values in main a = %d, b = %d\n”,a,b);

printf(“After swapping values in main a = %d, b = %d\n”,a,b);

void swap (int a, int b)

{

int temp;

temp = a;

a=b;

b=temp;

printf(“After swapping values in function a = %d, b = %d\n”,a,b);

getch();

}

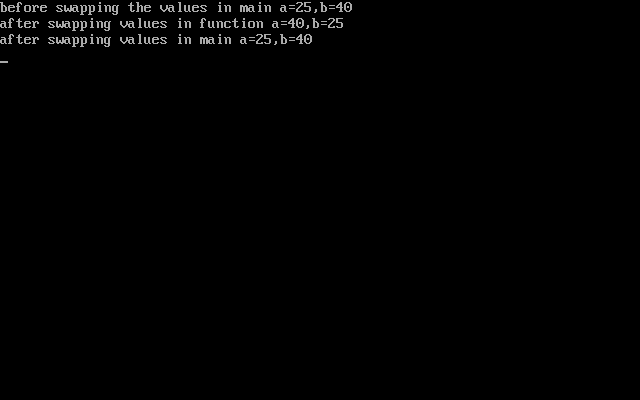
Expected Output:

Before swapping the values in main a = 25, b = 40

After swapping values in function a = 40, b = 25

After swapping values in main a = 25, b = 40

Actual output:



Result: Above program has been executed successfully.

## Call by reference

Experiment No.16 Roll No.1872062

Date: Page no.

Aim: To determine the swapping of two numbers using call by reference

Algorithm:

Step 1: start

Step 2: initializing a=10 B and b=20

Step 3: printing the a,b values using swapping

Step 4: calling the function swapp by passing the a,b values tem, \*a, \*b

Step 5: swapping the a, b values using temp

Step 6: printing the a,b values after r swapping the values in the function

Step 7: printing the a,b values after swapping the values in the main

Step 8: stop

Source code:

#include<stdio.h>

#include<conio.h>

void swap(int \*, int \*);

void main()

{

int a = 10;

int b = 20;

printf(“Before swapping the values in main a = %d, b = %d\n”,a,b);

printf(“After swapping values in main a = %d, b = %d\n”,a,b);

void swap (int \*a, int \*b)

{

int temp;

temp = \*a;

\*a=\*b;

\*b=temp;

printf(“After swapping values in function a = %d, b = %d\n”,\*a,\*b);

getch();

}

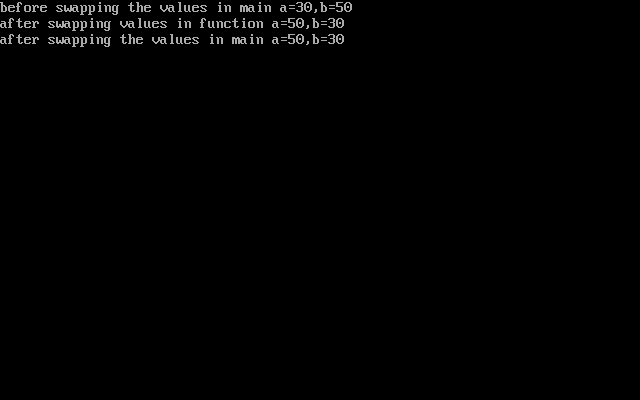
Expected output:

Before swapping the values in main a = 30, b = 50

After swapping values in function a = 50, b = 30

After swapping values in main a = 50,b = 30

Actual output:



Result: Above program has been executed successfully.

## Armstrong numbers

Experiment No.17 Roll No.1872062

Date: Page no.

Aim: To write a c program to find all armstrong numbers in range of 0 and 999.

Algorithm:

Step 1: Start.

Step 2: declare num , i , rem, cube, n, sum =0.

Step 3: enter range from 0 to 999

Step 4: for(i=0;i<=n;i++)

{

num=I; sum=0;

while(num!=0)

{

rem=num%10; cube=pow(rem,3);

sum=sum+cube; num=num/10;

}

If ( num == i ) is true go to step 5

Step 5: print amstrong numbers are:

Step 6: Stop.

Source code:

#include<stdio.h>

#include<conio.h>

#include<math.h>

void main()

{

int n,i,rem,cube,num,sum=0;

clrscr();

printf("\n enter the range from 0 to:");

scanf("%d",&n);

printf("\n Amstsrong numbers are:");

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

{

num=i;

sum=0;

while(num!=0)

{

rem=num%10;

cube=pow(rem,3);

sum=sum+cube;

num=num/10;

}

if(sum==i)

printf("\t%d",i);

}

getch();

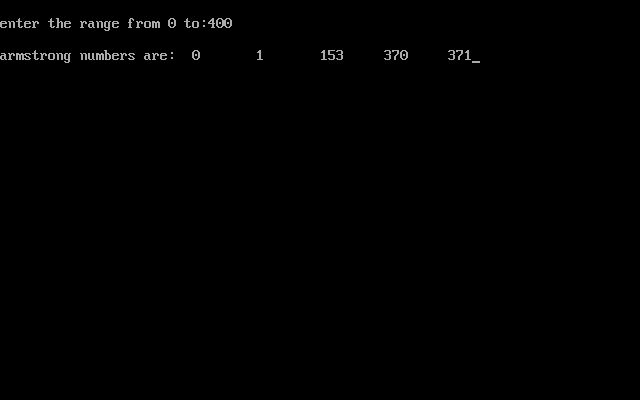
}

Expected output:

Enter the range from 0 to: 400

Armstrong numbers are: 0 1 153 370 371

Actual output:



Result: Above program has been executed successfully.

## Matrix operations

Experiment No. 18 Rollno.1872062

Date: Page no.

Aim: To write a c program to perform addition, subtraction, multiplication of matrices, transpose of a matrix, to find whether a given matrix is magic square or not, inverse of a matrix and to print upper and lower triangle of matrix using switch case.

Algorithm:

Step 1: Start.

Step 2: declare c,func1(),func2(),func3(),func4(),func5(),func6() and func7().

Step 3: use switch case to perform seven operations.

Step 4: to perform addition operation enter your choice as 1.

Step 5: to perform subtraction operation enter your choice as 2.

Step 6: to perform multiplication operation enter your choice as 3.

Step 7: to perform transpose of a matrix enter your choice as 4.

Step 8: find whether a given matrix is magic square or not, enter your choice as 5.

Step 9: to perform inverse of a matrix, enter your choice as 6.

Step 10: to print upper and lower triangle of matrix, enter your choice as 7.

Step 11: Stop.

Source code:

#include<stdio.h>

#include<conio.h>

void main()

{

int c;

void func1();

void func2();

void func3();

void func4();

void func5();

void func6();

void func7();

clrscr();

printf("\n-:Matrix Manipulation functions :");

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

printf("\n Matrix addition:1");

printf("\n matrix subtraction:2");

printf("\n matrix multiplication:3");

printf("\n transpose of a matrix matrix :4");

printf("\n matrix is magic square or not:5");

printf("\n inverse of a matrix:6");

printf("\n print the upper and lower triangle of matrix:7");

printf("\n enter your choice :");

scanf("%d",&c);

switch(c)

{

case 1:func1();

break;

case 2:func2();

break;

case 3:func3();

break;

case 4:func4();

break;

case 5:func5();

break;

case 6:func6();

break;

case 7:func7();

break;

default:

printf("\n invalid choice");

}

getch();

}

void func1()

{

int i, j, a[10][10], b[10][10], c[10][10];

int row1, col1, row2, col2;

printf("\nEnter the number of Rows of a: ");

scanf("%d", &row1);

printf("\nEnter the number of Cols of a : ");

scanf("%d", &col1);

printf("\nEnter the number of Rows of b : ");

scanf("%d", &row2);

printf("\nEnter the number of Columns of b : ");

scanf("%d", &col2);

if (row1 != row2 || col1 != col2)

{

printf("\nOrder of two matrices is not same ");

}

else

{

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

for (i = 0; i < row1; i++)

{

for (j = 0; j < col1; j++)

{

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

}

}

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

for (i = 0; i < row2; i++)

for (j = 0; j < col2; j++)

{

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

}

for (i = 0; i < row1; i++)

for (j = 0; j < col1; j++)

{

c[i][j] = a[i][j] + b[i][j];

}

printf("\nThe Addition of two Matrices is : \n");

for (i = 0; i < row1; i++)

{

for (j = 0; j < col1; j++)

{

printf("%d\t", c[i][j]);

}

printf("\n");

}

}

}

void func2()

{

int i, j, a[10][10], b[10][10], c[10][10];

int row1, col1, row2, col2;

printf("\nEnter the number of Rows of a: ");

scanf("%d", &row1);

printf("\nEnter the number of Cols of a : ");

scanf("%d", &col1);

printf("\nEnter the number of Rows of b : ");

scanf("%d", &row2);

printf("\nEnter the number of Columns of b : ");

scanf("%d", &col2);

if (row1 != row2 || col1 != col2)

{

printf("\nOrder of two matrices is not same ");

}

else

{

for (i = 0; i < row1; i++)

{

for (j = 0; j < col1; j++)

{

printf("Enter the Element a[%d][%d] : ", i, j);

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

}

}

for (i = 0; i < row2; i++)

for (j = 0; j < col2; j++)

{

printf("Enter the Element b[%d][%d] : ", i, j);

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

}

for (i = 0; i < row1; i++)

for (j = 0; j < col1; j++)

{

c[i][j] = a[i][j] - b[i][j];

}

printf("\nThe substraction of two Matrices is : \n");

for (i = 0; i < row1; i++)

{

for (j = 0; j < col1; j++)

{

printf("%d\t", c[i][j]);

}

printf("\n");

}

}

}

void func3()

{

int m, n, p, q, c, d, k, sum = 0;

int first[10][10], second[10][10], multiply[10][10];

printf("Enter number of rows and columns of first matrix\n");

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

printf("Enter elements of first matrix\n");

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

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

scanf("%d", &first[c][d]);

printf("Enter number of rows and columns of second matrix\n");

scanf("%d%d", &p, &q);

if (n != p)

printf("The matrices can't be multiplied with each other.\n");

else

{

printf("Enter elements of second matrix\n");

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

for (d = 0; d < q; d++)

scanf("%d", &second[c][d]);

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

{

for (d = 0; d < q; d++)

{

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

{

sum = sum + first[c][k]\*second[k][d];

}

multiply[c][d] = sum;

sum = 0;

}

}

printf("Product of the matrices:\n");

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

{

for (d = 0; d < q; d++)

printf("%d\t", multiply[c][d]);

printf("\n");

}

}

}

void func4()

{

int m, n, c, d, matrix[10][10], transpose[10][10];

printf("Enter the number of rows and columns of matrix\n");

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

printf("Enter elements of the matrix\n");

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

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

scanf("%d", &matrix[c][d]);

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

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

transpose[d][c] = matrix[c][d];

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

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

{

for (d = 0; d < m; d++)

printf("%d\t", transpose[c][d]);

printf("\n");

}

}

void func5()

{

int size = 3;

int matrix[3][3];

int row, column = 0;

int sum, sum1, sum2;

int flag = 0;

printf("\nEnter matrix : ");

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

{

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

scanf("%d", &matrix[row][column]);

}

printf("Entered matrix is : \n");

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

{

printf("\n");

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

{

printf("\t%d", matrix[row][column]);

}

}

//For diagonal elements

sum = 0;

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

{

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

{

if (row == column)

sum = sum + matrix[row][column];

}

}

//For Rows

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

{

sum1 = 0;

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

{

sum1 = sum1 + matrix[row][column];

}

if (sum == sum1)

flag = 1;

else

{

flag = 0;

break;

}

}

//For Columns

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

{

sum2 = 0;

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

{

sum2 = sum2 + matrix[column][row];

}

if (sum == sum2)

flag = 1;

else

{

flag = 0;

break;

}

}

if (flag == 1)

printf("\nMagic square");

else

printf("\nNo Magic square");

}

void func6()

{

int mat[3][3], i, j;

float determinant = 0;

printf("Enter elements of matrix row wise:\n");

for(i = 0; i < 3; i++)

for(j = 0; j < 3; j++)

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

printf("\nGiven matrix is:");

for(i = 0; i < 3; i++)

{

printf("\n");

for(j = 0; j < 3; j++)

printf("%d\t", mat[i][j]);

}

//finding determinant

for(i = 0; i < 3; i++)

determinant = determinant + (mat[0][i] \* (mat[1][(i+1)%3] \* mat[2][(i+2)%3] - mat[1][(i+2)%3] \* mat[2][(i+1)%3]));

printf("\n\ndeterminant: %f\n", determinant);

printf("\nInverse of matrix is: \n");

for(i = 0; i < 3; i++)

{

for(j = 0; j < 3; j++)

printf("%.2f\t",((mat[(j+1)%3][(i+1)%3] \* mat[(j+2)%3][(i+2)%3]) - (mat[(j+1)%3][(i+2)%3] \* mat[(j+2)%3][(i+1)%3]))/ determinant); printf("\n");

}

}

void func7()

{

int a[10][10],i,j,row\_size,col\_size;

printf("\nEnter the row and column size of matrix");

scanf("%d %d",&row\_size,&col\_size);

printf("\nEnter the matrix elements->\n");

for(i=0;i<row\_size;i++)

for(j=0;j<col\_size;j++)

scanf(“%d”,a[i][j]);

printf(“\n entered matrix is\n”);

for(i=0;i<row\_size;i++)

{

printf(“\n”);

for(j=0;j<col\_size;j++)

printf(“%d\t”,a[i][j]);

}

printf("\nThe lower diagonal elements of the matrix is\n");

for(i=0;i<row\_size;i++)

{

Printf(“\n”);

for(j=0;j<col\_size;j++)

{

If(i>j)

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

}

}

printf("\nThe upper diagonal elements of the matrix is\n");

for(i=0;i<row\_size;i++)

{

printf(“\n”);

for(j=0;j<col\_size;j++)

{

If(i<j)

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

else

printf(“”\t);

}

}

}

Expected output:

-:Matrix Manipulation functions :

--------------------------

Matrix addition:1

matrix subtraction:2

matrix multiplication:3

transpose of a matrix matrix :4

matrix is magic square or not:5

inverse of a matrix:6

print the upper and lower triangle of matrix:7

enter your choice :1

Enter the number of Rows of a: 1

Enter the number of Cols of a : 2

Enter the number of Rows of b : 2

Enter the number of Columns of b : 1

Order of two matrices is not same.

-:Matrix Manipulation functions :

--------------------------

Matrix addition:1

matrix subtraction:2

matrix multiplication:3

transpose of a matrix matrix :4

matrix is magic square or not:5

inverse of a matrix:6

print the upper and lower triangle of matrix:7

enter your choice :2

Enter the number of Rows of a: 2

Enter the number of Cols of a : 2

Enter the number of Rows of b : 2

Enter the number of Columns of b : 2

Enter the Element a[0][0] : 6

Enter the Element a[0][1] : 3

Enter the Element a[1][0] : 5

Enter the Element a[1][1] : 0

Enter the Element b[0][0] : 2

Enter the Element b[1][0] : 7

Enter the Element b[1][1] : 4

The substraction of two Matrices is :

4 2

-2 -4

-:Matrix Manipulation functions :

--------------------------

Matrix addition:1

matrix subtraction:2

matrix multiplication:3

transpose of a matrix matrix :4

matrix is magic square or not:5

inverse of a matrix:6

print the upper and lower triangle of matrix:7

enter your choice :3

Enter number of rows and columns of first matrix 2 2

Enter elements of first matrix 2 5 7 1

Enter number of rows and columns of second matrix 2 2

Enter elements of second matrix 8 4 -3 6

Product of the matrices:

1 38

53 34

-:Matrix Manipulation functions :

--------------------------

Matrix addition:1

matrix subtraction:2

matrix multiplication:3

transpose of a matrix matrix :4

matrix is magic square or not:5

inverse of a matrix:6

print the upper and lower triangle of matrix:7

enter your choice :4

Enter the number of rows and columns of matrix 3 3

Enter elements of the matrix

2 4 -1

5 9 0

6 3 7

Transpose of the matrix:

2 5 6

4 9 3

-1 0 7

-:Matrix Manipulation functions :

--------------------------

Matrix addition:1

matrix subtraction:2

matrix multiplication:3

transpose of a matrix matrix :4

matrix is magic square or not:5

inverse of a matrix:6

print the upper and lower triangle of matrix:7

enter your choice :5

Enter matrix : 10 3 8

5 7 9

6 11 4

Entered matrix is :

10 3 8

5 7 9

6 11 4

Magic square

-:Matrix Manipulation functions :

--------------------------

Matrix addition:1

matrix subtraction:2

matrix multiplication:3

transpose of a matrix matrix :4

matrix is magic square or not:5

inverse of a matrix:6

print the upper and lower triangle of matrix:7

enter your choice :6

Enter elements of matrix row wise: 1 2 3 0 1 4 5 6 0

Given matrix is:

1 2 3

0 1 4

5 6 0

determinant: 1.000000

Inverse of matrix is:

-24.00 18.00 5.00

20.00 -15.00 -4.00

-5.00 4.00 1.00

-:Matrix Manipulation functions :

--------------------------

Matrix addition:1

matrix subtraction:2

matrix multiplication:3

transpose of a matrix matrix :4

matrix is magic square or not:5

inverse of a matrix:6

print the upper and lower triangle of matrix:7

enter your choice :7

Enter the row and column size of matrix2 2

Enter the matrix elements-> 6 8 9 1

The entered matrix is

6 8

9 1

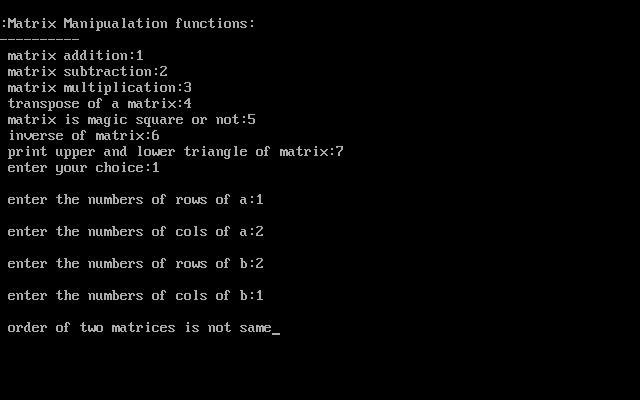
The lower diagonal elements of the matrix is

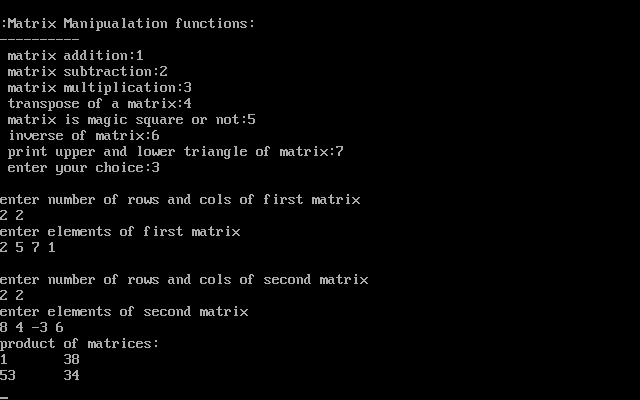
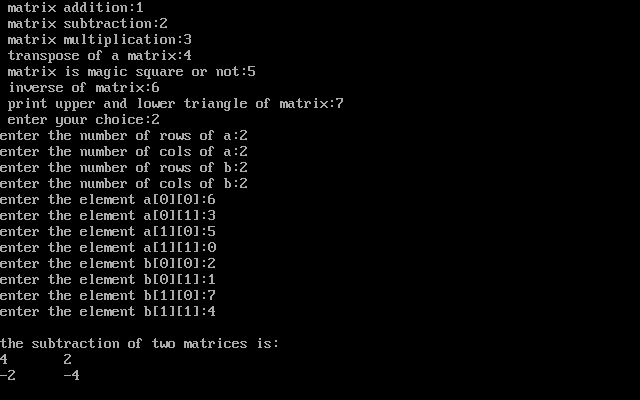
9

The upper diagonal elements of the matrix is

8

Actual output:





## Screenshot (78).png

## Screenshot (79).png

## Screenshot (81).png

## Screenshot (82).png

Result: Above program has been executed successfully.

## Sorting of strings