

|  |
| --- |
| C.P.N.M. LAB REPORT |
|  |
| MARCH 18TH  BCSE FIRST YEAR FIRST SEMESTER  Authored by: SOHAM CHOWDHURY |

|  |
| --- |
| **CPNM LAB ASSIGNMENT REPORT**  BCSE FIRST YEAR FIRST SEMESTER 2021-2022  NAME-SOHAM CHOWDHURY  DEPARTMENT-COMPUTER SCIENCE AND ENGINEERING  SECTION-A3.  ROLL NO-002110501145. |
| CONTENTS-ASSIGNMENT 1,2,3,4,5,6,7. |

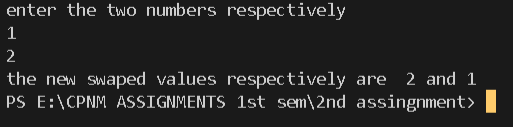
# ASSIGNMENT 2

1.Write a C program that reads two values from the keyboard, swaps their values and prints out the result.

Here, we performed swapping without using third variable.

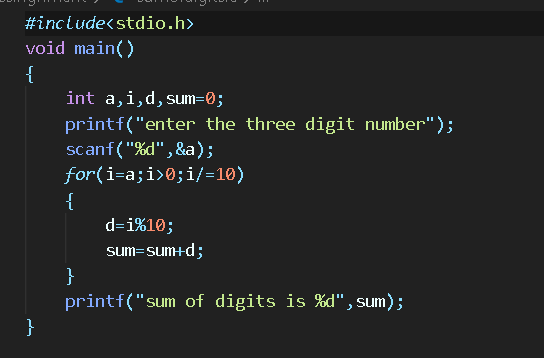
# 

OUTPUT-



2.If a three-digit integer is input through the keyboard, write a program to calculate the sum of its digits. (Hint: Use the modulo operator ‘%’)

HERE,we used the logic of using digit extraction using modulus operator then storing the digits sum in a variable.

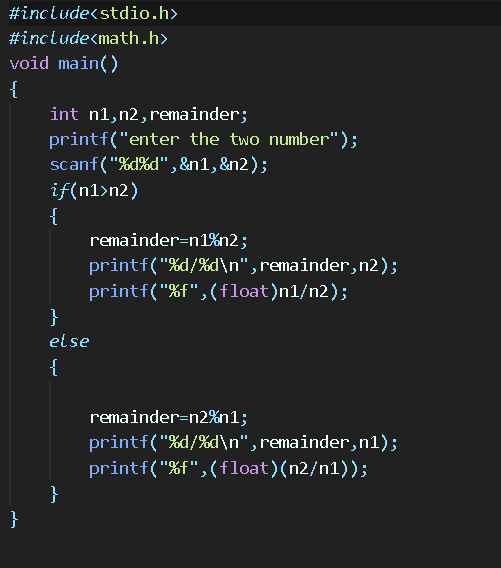


OUTPUT-

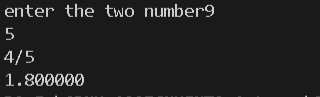


3. Input two integer numbers and divide the larger number by the smaller one. Then display the result using printf() function as a fractional number first and then as a real valued number. (Example: 9 divided by 5 shall yield “ 4/5” and “1.8” respectively. )

HERE,we found the number which is smaller and greater among the two numbers given by the user as input and then printed the division resultant in float and fraction was obtained by the remainder remainder found by using modulus operation among the two numbers.

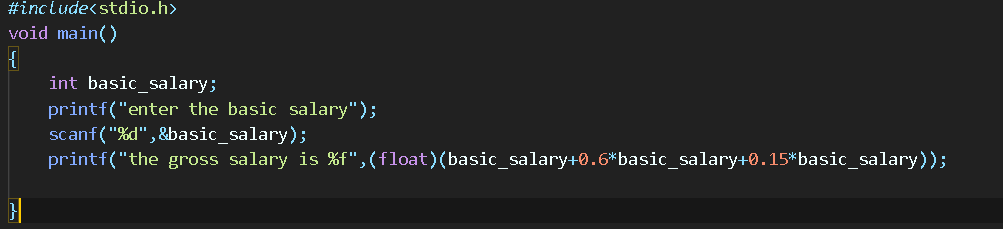


OUTPUT-



4. Write a C program which accepts basic salary as input and prints the gross salary, which is the sum of the basic, dearness allowance (60% of basic salary), and house rent allowance (15% of basic salary).

Here,we just took the basic salary of the employee from the user and expressed dearness allowance in terms of the basic salary then just printed it.

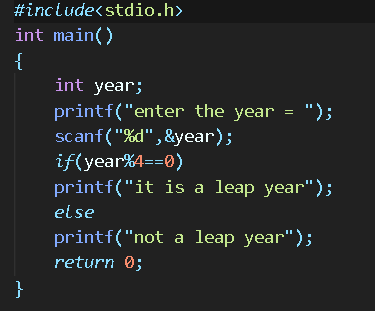


OUTPUT-



5. Any year is input through the keyboard. Write a program to determine whether the year is a leap year or not. (Hint: Use the % (modulus) operator)

WE TAKE THE YEAR AS INPUT FROM THE USER THEN IF IT IS DIVISIBLE BY 4 THEN LEAP YEAR ELSE NOT A LEAP YEAR.

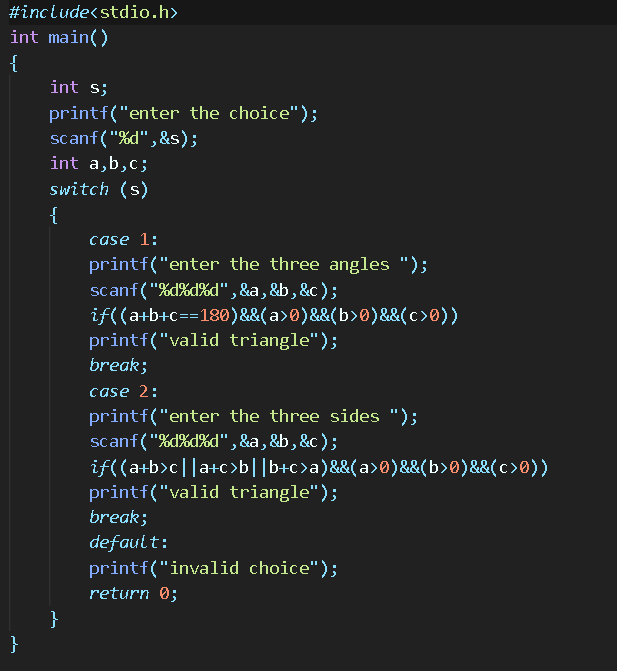


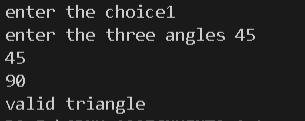
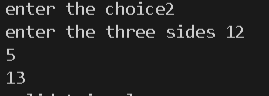
OUTPUT-



6. Write a program to check whether a triangle is valid or not, when (i) the three angles of the triangle are entered through the Keyboard (ii) three sides of the triangle are entered through the keyboard.

WE TAKE SIDES OR ANGLES AS INPUT AND IF THE SUM OF ALL ANGLES IS EQUAL TO 180 OR IF SUM OF TWO SIDES IS GREATER THAN THE THIRD THEN THE TRIANGLE EXISTS.

OUTPUT-

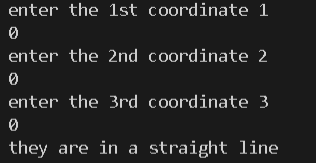


7. Given three points (x1, y1), (x2, y2) and (x3, y3), write a program to check if all the three points fall on one straight line.

TO CHECK WHETER THREE POINTS LIE IN THE SAME LINE I CHECKED THE SLOPE BETWEEN TWO POINTS AND COMPARED IT WITH THE SLOPE BETWEEN AND OF THE ONE POINTS WITH THE THIRD POINT.

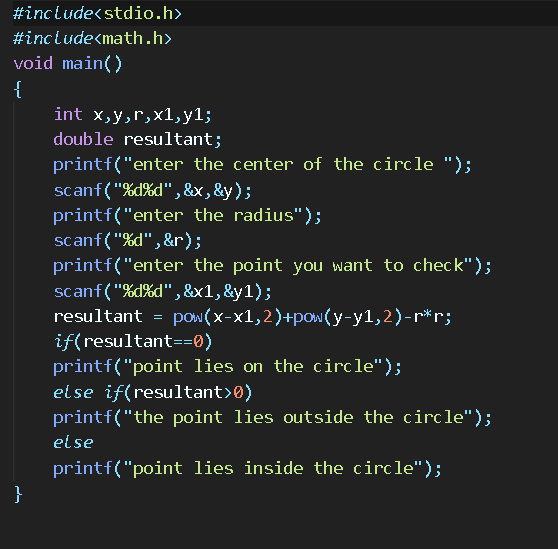


OUTPUT-

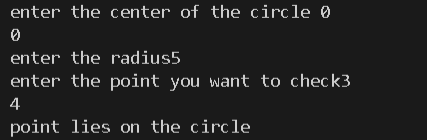


8. Given the coordinates (x, y) of a center of a circle and its radius, write a program which will determine whether a point lies inside the circle, on the circle or outside the circle. (Hint: #include . Use sqrt( ) and pow( ) functions)

Here I use the distance formula between an arbitrary point defined by the user and the centre of the circle which was also defined by the user and if it was greater than radius of the circle then point lies outside,if equal to radius then point lies on the circle,else inside the circle.

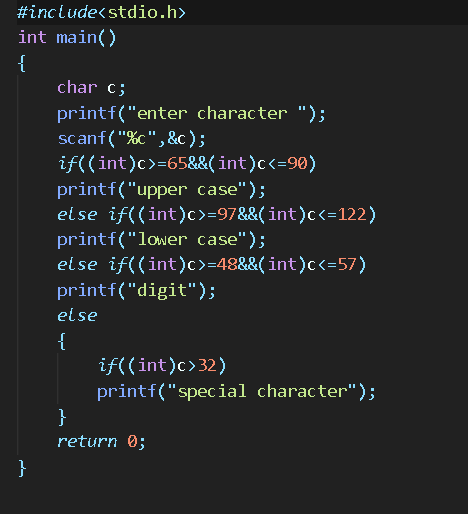


OUTPUT-

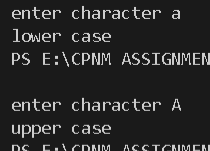


9. Any character is entered through the keyboard, write a program to determine whether the character entered is a capital letter, a small case letter, a digit or a special symbol.

HERE I USED THE CONCEPT OF ASCII VALUES TO FIND OUT THE TYPE OF CHARACTERS GIVEN AS INPUT FROM THE USER.

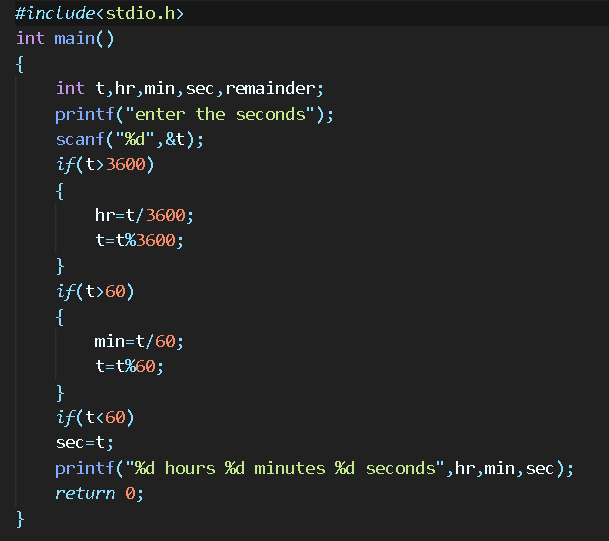


OUTPUT-



10. Given as input an integer number of seconds, write a program to print as output the equivalent time in hours, minutes and seconds. Recommended output format is something like 7322 seconds is equivalent to 2 hours 2 minutes 2 seconds

WE FIRST CHECK WHETHER THE SECONDS GIVEN BY USER IS EQUIVALENT TO HOURS MINUTES OR SECOND THEN CALCULATE THE NUMBER OF HOURS THEN SECONDS THEN MINUTES AND PRINT THEM.



OUTPUT-



# ASSIGNMENT 3

1. Write a C program which accepts a number n and prints a. all integers divisible by n between 1 and 100 where value of n is provided by the user. b. all prime numbers between 1 and n. c. all prime factors of n. d. octal equivalent of n e. sum of digits. f. factorial of n. g. reverse of n.

HERE, a.i take a loop from one to hundred and take n from the user and if i%n==0 then print the number.

b.take n as input from user and then take a loop from 1 to n with variable I and check is I is prime or not,prime number is divisible only by one and itself.

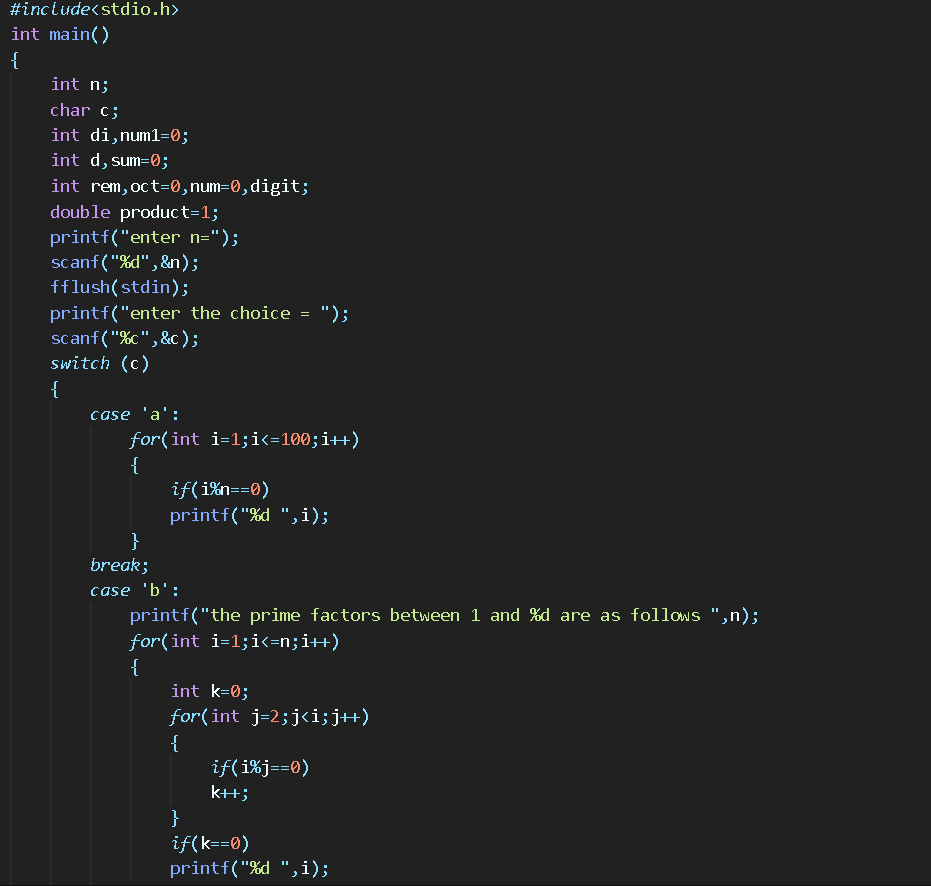
c.take a number as input find its factors with the help of loop, check whether the factors are prime or not,if prime print them.

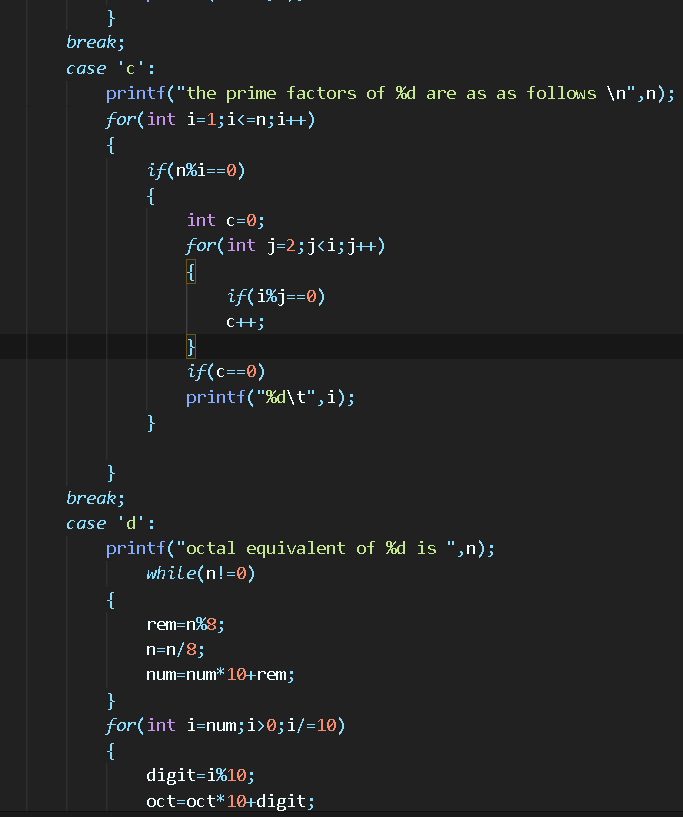
d.converted the decimal number inputted by user to octal number.

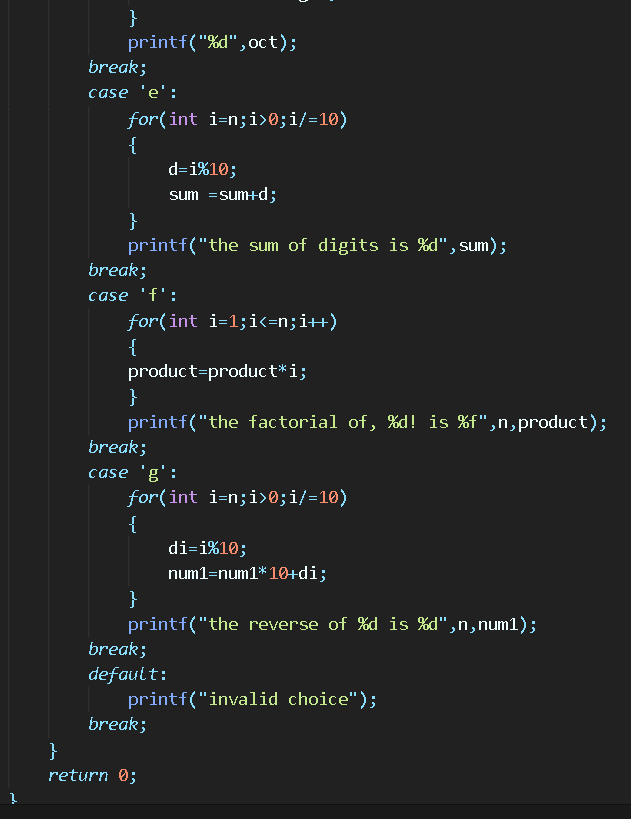
e.we used the logic of using digit extraction using modulus operator then storing the digits sum in a variable.

f.using loop from 1 to n and storing their product gives the factorial of a number.

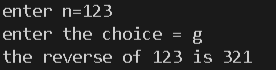
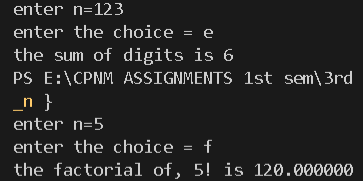
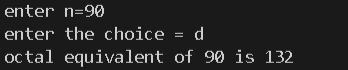
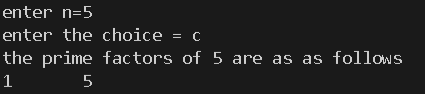
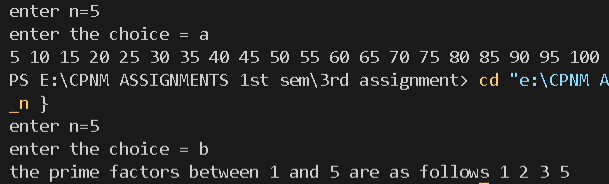
g.reversed the number by digit extraction then multiplying them with 10.







OUTPUT-



1. Write a C program to find out the sum of the following series. a. S=1+2+3+4+ ... +n b. S=1.2+2.3+3.4+4.5+ ... +n.(n+1) c. S=1!+2!+3!+4!+ ... +n! d. S = 1@ + 2@ + 3@ + 4@ + … + n@ where, n@ is the sum of all factors of n. Example: 6@ = 1+2+3+6 = 12

My main objective here was to print all the series using loop and expressing each general term of each series in terms of the looping variable then storing their sum with each iteration of the loop.

Program:

*#include*<stdio.h>

int factorial(int a)

{

*if*(a==0||a==1)

*return* 1;

*else*

*return* a\*factorial(a-1);

}

int sumoffact(int a)

{

    int sum1=0;

*for*(int i=1;i<=a;i++)

    {

*if*(a%i==0)

        sum1=sum1+i;

    }

*return* sum1;

}

int main()

{

    char c;

    printf("enter choice=");

    scanf("%c",&c);

    int n,sum=0;

    fflush(stdin);

    printf("enter n=");

    scanf("%d",&n);

*switch* (c)

    {

*case* 'a':

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

        {

            sum=sum+i;

        }

        printf("the sum is %d",sum );

*break*;

*case* 'b':

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

        {

            sum=sum+i\*(i+1);

        }

        printf("the sum is %d",sum );

*break*;

*case* 'c':

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

        {

            int k=factorial(i);

            sum=sum+k;

        }

        printf("the sum is %d",sum);

*break*;

*case* 'd':

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

        {

            int b=sumoffact(i);

            sum=sum+b;

        }

        printf("the sum is %d",sum);

*break*;

*default*:

        printf("invalid choice ");

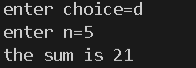
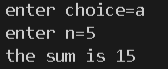
*break*;

    }

*return* 0;

}

Output:



1. Write a program to generate all combinations of digit 1, 2 and 3 using a for loop.

I used the logic of three nested loops to generate all combinations of 1,2,3 such that none of them are same at the same time.

*#include*<stdio.h>

int main()

{

    printf("the combinations of digits 1,2,3 are as follows:-\n");

*for*(int i=1;i<=3;i++)

    {

*for*(int j=1;j<=3;j++)

        {

*for*(int k=1;k<=3;k++)

            {

*if*(i!=j&&j!=k&&i!=k)

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

            }

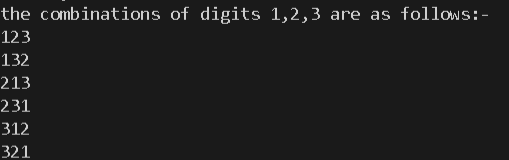
        }

    }

*return* 0;

}

Output:



1. Write a menu driven program to accept a number in any Number System [from Binary, Decimal, Octal, and Hex] and convert and display the same in any other amongst these.

Here my main logic was to input the number from the user along with its base and the base of the resultant number.

First I converted the number to decimal then I converted it to the required base if required.

*#include*<stdio.h>

*#include*<string.h>

*#include*<stdlib.h>

int value(char c)

{

*if*(c>='0'&&c<='9')

*return* (int)c-'0';

*else*

*return* (int)c -'A'+10;

}

int any\_to\_dec(char \*str, int base)

{

    int len = strlen(str);

    int power = 1;

    int num = 0;

    int i;

*for*(i=len-1;i>=0;i--)

    {

        num += value(str[i]) \* power;

        power = power \* base;

    }

*return* num;

}

char reVal(int num)

{

*if*(num >=0 && num <=9)

*return* (char)(num + '0');

*else*

*return* (char)(num - 10 + 'A');

}

char\* dec\_to\_any( int inputNum, int base)

{

    static char res[100];

    int index=0;

*while*(inputNum > 0)

    {

        res[index++] = reVal(inputNum % base);

        inputNum /= base;

    }

    res[index] = '\0';

    int len = strlen(res);

    int i;

*for*(i=0;i<len/2;i++)

    {

        char temp = res[i];

        res[i] = res[len-i-1];

        res[len-i-1] =temp;

    }

*return* res;

}

int main()

{

    int ibase,fbase;

    char \* ptr;

    printf("enter the initial base=");

    scanf("%d",&ibase);

    printf("enter the final base=");

    scanf("%d",&fbase);

    char str[100];

    fflush(stdin);

    printf("enter the number=");

    gets(str);

    char\* str1;

    int x,y,z;

*if*(ibase==fbase)

    puts(str);

*else*

    {

        int ch;

*if*(ibase == 10)

        ch=1;

*else* *if*(fbase == 10)

        ch=2;

*else*

        ch=3;

*switch*(ch)

        {

*case* 1:

            x=atoi(str);

            str1=dec\_to\_any(x,fbase);

            printf("the resultant number is=");

            puts(str1);

*break*;

*case* 2:

            y=any\_to\_dec(str,ibase);

            printf("the resultant number is=%d",y);

*break*;

*case* 3:

            z=any\_to\_dec(str,ibase);

            ptr=dec\_to\_any(z,fbase);

            printf("the resultant number is=");

            puts(ptr);

*break*;

*default*:

            printf("invalid input");

*break*;

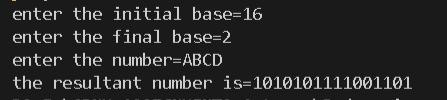
        }

    }

*return* 0;

}

Output:



1. Write a program named SINE to find the sine of an angle. The angle and its unit (degree, radian or grade) should be provided as command line arguments. For the units, short forms as d/D (for degree), r/R (for radian) or g/G (for grade) may be used. The program should use the series sin 𝑠𝑖𝑛 (𝑥) = 𝑥 − , 𝑥 3 3! + 𝑥 5 5! − ⋯ for evaluation. Take care of negative angles and angles in all the quadrants.

Here I took the input through command line arguments as per mentioned in the question 2nd and 3rd arguments were value and unit respectively

*#include*<stdio.h>

*#include*<math.h>

*#include*<stdlib.h>

*#include*<string.h>

*#define* PI 3.14159265

double factorial(int n)

{

*if*(n==0||n==1)

*return* 1;

*else*

*return* n\*factorial(n-1);

}

int main(int argc,char const\* argv[])

{

*if*(argc==3)

    {

        int x;

        double sum=0;

        float rad;

        x = atoi(argv[1]);

        printf("the given angle value is=%d\n",x);

*if*(\*argv[2]=='r'||\*argv[2]=='R')

        rad=x;

*else* *if*(\*argv[2]=='d'||\*argv[2]=='D')

        rad=PI\*x/180;

*else* *if*(\*argv[2]=='g'||\*argv[2]=='G')

        rad=x\*0.015708;

*for*(int i=1;i<=100;i++)

        {

            sum=sum+(double)(pow(-1,i-1)\*pow(rad,2\*i-1)/factorial(2\*i-1));

        }

        printf("sin(%d%s) = %0.3f",x,argv[2],sum);

    }

*else*

    {

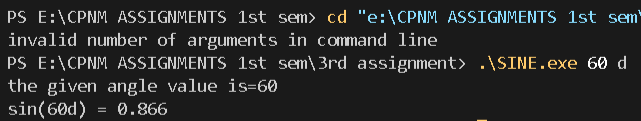
        printf("invalid number of arguments in command line");

    }

*return* 0;

}

Output:



1. Write a C program to print the first n numbers of the Fibonacci sequence. The Fibonacci sequence is constructed by adding the last two numbers of the sequence so far to get the next number in the sequence. The first and second numbers of the sequence are defined as 0 and 1. We get: 0, 1, 1, 2, 3, 5, 8, 13, 21…

Using loop and assuming first element to be 0 and 1 , I printed the Fibonacci series in which the n th term is the sum of n-1th term and n-2 th term.

*#include*<stdio.h>

int main()

{

    int n,sum;

    printf("enter n=");

    scanf("%d",&n);

    int a=0,b=1;

*if*(n>=2)

    {

        printf("%d,%d",a,b);

*for*(int i=1;i<n-1;i++)

        {

            sum=a+b;

            printf(",%d",sum);

            a=b;

            b=sum;

        }

    }

*else* *if*(n==1)

    printf("%d",a);

*else*

    printf("invalid number of terms");

*return* 0;

}

Output:



1. Write a program to print out all Armstrong numbers between 1 and 500. If the sum of cubes of each digit of the number is equal to the number itself, then the number is called an Armstrong number. For example, 153 = ( 1 3 ) + ( 5 3 ) + ( 3 3 ).

Using loop I extracted each and every digit of the number given by the user then stored their cube in the sum variable and if it is equivalent to the number itself then it is an Armstrong number.

*#include*<stdio.h>

*#include*<math.h>

int main()

{

    int d;

    printf("the armstrong numbers between 1 and 500 are as follows :-\n");

*for*(int i=1;i<=500;i++)

    {

        int sum=0;

*for*(int j=i;j>0;j=j/10)

        {

            d=j%10;

            sum=sum+d\*d\*d;

        }

*if*(sum==i)

        printf("%d\t",i);

    }

*return* 0;

}

Output:



1. Write a C program which prints the first 10 happy numbers. If you iterate the process (assume maximum 100 iterations) of summing the squares of the decimal digits of a number and if the process terminates in 1, then the original number is called a Happy number. For example 7 is a happy number as 7 → 49 → 97 → 130 → 10 → 1.

Here I used the principles of digit extraction then storing sum of the squares of the digit in another variable and made that the new number until it becomes single digit and if itn is one then the number is happy.

*#include*<stdio.h>

*#include*<stdio.h>

*#include*<math.h>

void main()

{

    int n=1,num=10,d;

    printf("the 1st 10 happy numbers are as follows:-\n");

*for*(int c=0;c<10;)

    {

        num=n;

*do*

        {

            int sum=0;

*for*(int i=num;i>0;i=i/10)

            {

                d=i%10;

                sum=sum+d\*d;

            }

            num=sum;

        }*while* (num>9);

*if*(num==1)

        {

            printf("%d\t",n);

            c++;

        }

        n++;

    }

}

Output:



1. An important property of square numbers: If a natural number is a square number, then it has to be the sum of Successive Odd Numbers starting from 1. For example: Perfect Square Sum of Odd

Sum of first n odd numbers is equal to the square of the number n.

*#include*<stdio.h>

*#include*<math.h>

int main()

{

    int n,c=0,sum=0;

    printf("enter the perfect square number = ");

    scanf("%d",&n);

*for*(int i=1;i<n;i=i+2)

    {

        sum=sum+i;

        c++;

*if*(sum==n)

*break*;

    }

    printf("the square root of %d is = %d",n,c);*//sum of n consecutive numbers = n\*n*

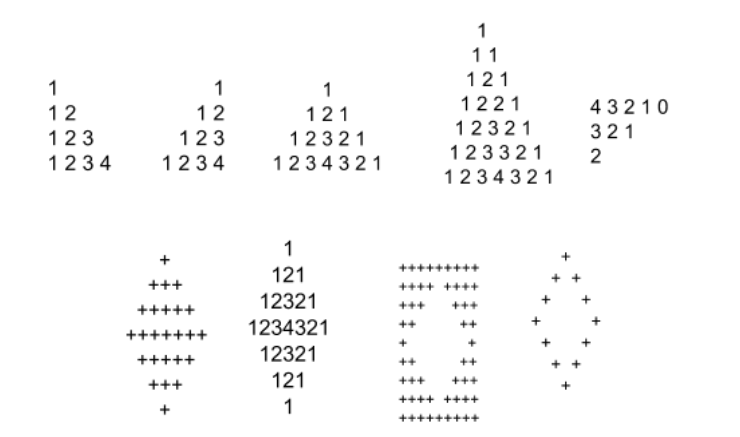
*return* 0;

}

Ouput:



1. W rite a C program that prints the following patterns for the input of n = 4. T he value of n is input by the user



In all the 9 types of patterns I devided the pattern into multiple parts and I used the looping variable i from row tracking j for column tracking also used spaces and starts where it was necessary. PROGRAMS ARE IN ORDER OF TOP TO BOTTOM AND LEFT TO RIGHT SIMULTANEOUSLY.

Program:

*#include*<stdio.h>

*#include*<math.h>

int main()

{

    int n;

    printf("enter n=");

    scanf("%d",&n);

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

    {

*for*(int j=1;j<=i;j++)

        {

            printf("%d",j);

        }

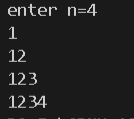
        printf("\n");

    }

*return* 0;

}

Output:



Program:

*#include*<stdio.h>

*#include*<math.h>

int main()

{

    int n;

    printf("enter n=");

    scanf("%d",&n);

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

    {

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

        {

            printf(" ");

        }

*for*(int j=1;j<=i;j++)

        {

            printf("%d",j);

        }

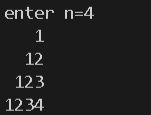
        printf("\n");

    }

*return* 0;

}

Output:



Program:

*#include*<stdio.h>

*#include*<math.h>

int main()

{

    int n;

    printf("enter n=");

    scanf("%d",&n);

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

    {

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

        {

            printf(" ");

        }

*for*(int j=1;j<=i;j++)

        {

            printf("%d",j);

        }

*if*(i>=2)

        {

*for*(int k=i-1;k>=1;k--)

            {

                printf("%d",k);

            }

        }

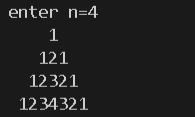
        printf("\n");

    }

*return* 0;

}

Output:



Program:

*#include*<stdio.h>

*#include*<math.h>

int main()

{

    int n;

    printf("enter n=");

    scanf("%d",&n);

    int k=n-1;

*for*(int i=1;i<=2\*n-1;i++)

    {

*if*(i<=n)

        {

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

            printf(" ");

*for*(int j=1;j<=2\*i-1;j=j+1)

            printf("\*");

        }

*else*

        {

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

            printf(" ");

*for*(int j=1;j<=2\*k-1;j++)

            printf("\*");

*if*(k==1)

*break*;

            k--;

        }

        printf("\n");

    }

*return* 0;

}

Output:



Program:

*#include*<stdio.h>

int main()

{

    int n;

    printf("enter n=");

    scanf("%d",&n);

    int b=0;

*for*(int i=1;i<=2\*n-1;i++)

    {

*if*(i<=n)

        {

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

            printf(" ");

*for*(int j=1;j<=i;j++)

            printf("%d",j);

*if*(i>=2)

            {

*for*(int k=i-1;k>=1;k--)

                printf("%d",k);

            }

        }

*else* *if*(i>n)

        {

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

            printf(" ");

*for*(int k=1;k<=2\*n-i;k++)

            printf("%d",k);

*for*(int c=2\*n-i-1;c>=1;c--)

            printf("%d",c);

        }

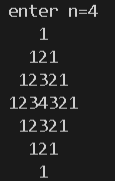
        printf("\n");

    }

*return* 0;

}

Output:



Program:

*#include*<stdio.h>

int main()

{

    int n,d=2;

    printf("enter n=");

    scanf("%d",&n);

    int c=n,f=2\*n-3;

*for*(int i=1;i<=2\*n+1;i++)

    {

*if*(i==1||i==2\*n+1)

            {

*for*(int j=1;j<=2\*n+1;j++)

                printf("\*");

            }

*if*(i>=2&&i<=n+1)

            {

*for*(int j=1;j<=c;j++)

                printf("\*");

*for*(int k=1;k<=2\*i-3;k++)

                printf(" ");

*for*(int j=1;j<=c;j++)

                printf("\*");

                c--;

            }

*if*(i>=n+2&&i<=2\*n)

            {

*for*(int j=1;j<=d;j++)

                printf("\*");

*for*(int j=1;j<=f;j++)

                printf(" ");

                f=f-2;

*for*(int j=1;j<=d;j++)

                printf("\*");

                d++;

            }

            printf("\n");

    }

*return* 0;

}

Output:



Program:

*#include*<stdio.h>

int main()

{

    int n;

    printf("enter n=");

    scanf("%d",&n);

    int d=2\*n-5;

*for*(int i=1;i<=2\*n-1;i++)

    {

*if*(i<=n)

        {

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

            {

                printf(" ");

            }

            printf("\*");

*for*(int j=1;j<=2\*i-3;j++)

            printf(" ");

*if*(i>=2&&i<=n)

            {

                printf("\*");

            }

        }

*else*

        {

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

            printf(" ");

            printf("\*");

*for*(int j=1;j<=d;j++)

            printf(" ");

            d=d-2;

*if*(i<2\*n-1)

            {

                printf("\*");

            }

        }

        printf("\n");

    }

*return* 0;

}

Output:



Program:

*#include*<stdio.h>

int main()

{

    int n,k;

    printf("enter n=");

    scanf("%d",&n);

*for*(int i=1;i<=2\*n-1;i++)

    {

*if*(i%2!=0)

        {

            int k=(i+1)/2;

*for*(int j=1;j<=n-k;j++)

            printf("  ");

*for*(int j=1;j<=k;j++)

            printf("%d ",j);

*for*(int j=k-1;j>=1;j--)

            printf("%d ",j);

        }

*else*

        {

            int d=i/2;

*for*(int j=2\*n-1-i;j>=1;j--)

            printf(" ");

*for*(int j=1;j<=d;j++)

            printf("%d ",j);

*for*(int j=d;j>=1;j--)

            printf("%d ",j);

        }

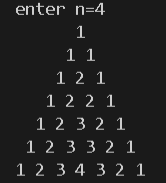
        printf("\n");

    }

*return* 0;

}

Output:



Program:

*#include*<stdio.h>

int main()

{

    int n,c,d;

    printf("enter n=");

    scanf("%d",&n);

    c=n;

    d=c;

*for*(int i=1;i<=n-1;i++)

    {

*for*(int j=1;j<=2\*n-1-2\*i;j++)

        {

            printf("%d",c);

            c--;

        }

        d--;

        c=d;

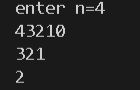
        printf("\n");

    }

*return* 0;

}

Output:



# ASSIGNMENT 4

1. Write a program in C to reverse the contents of the elements of an integer array.

I RAN A LOOP FROM INDEX 0 TO LENGTH/2 FOR AN ARRAY AND SWAPPED THE 1ST AND LAST 2ND AND SECOND LAST AND SO ON USING THIRD VARIABLE.

Program:

*#include*<stdio.h>

*#include*<stdlib.h>

int\* reverse(int l,int arr[])

{

    int temp;

*for*(int i=0;i<l/2;i++)

    {

        int temp=arr[i];

        arr[i]=arr[l-i-1];

        arr[l-i-1]=temp;

    }

*return* (arr);

}

int main()

{

    int\* arr,length;

    int \*ptr;

    printf("enter the length of the array");

    scanf("%d",&length);

    arr=(int\*)calloc(length,sizeof(int));

    printf("enter the elements of the array:-\n");

*for*(int i=0;i<length;i++)

    scanf("%d",&arr[i]);

    printf("the old array is :-\n");

*for*(int i=0;i<length;i++)

    printf("%d",arr[i]);

    printf("\n");

    ptr=reverse(length,arr);

    printf("the reversed array is:-\n");

*for*(int i=0;i<length;i++)

    printf("%d",ptr[i]);

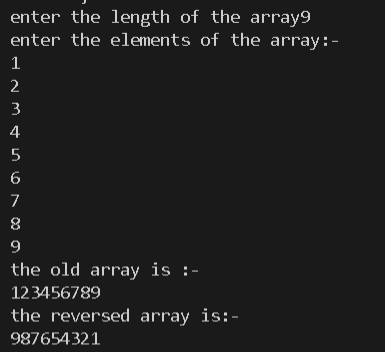
    free(arr);

    free(ptr);

*return* 0;

}

Output:



1. Write a program in C to read n number of values in an array. After that, count the total number of duplicate elements in that array. Then copy the elements except the duplicate elements of that array into another array and display this array in reverse order.

Using nested loop I find the repeating elements in the array and as soon as a duplicate is found I increase the counter and delete the duplicate element as well.

Program:

*#include*<stdio.h>

*#include*<stdlib.h>

int main()

{

    int \* arr=NULL;

    int n,d,c=0;

    printf("enter the number of elements=");

    scanf("%d",&n);

    arr=(int\*)calloc(n,sizeof(int));

    printf("enter the elements of the array");

    int l=n;

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

    {

        scanf("%d",&arr[i]);

    }

*for*(int i=0;i<l-1;i++)

    {

*for*(int j=i+1;j<l;j++)

        {

*if*(arr[i]==arr[j])

            {

                c++;

*for*(int k=j;k<l-1;k++)

                arr[k]=arr[k+1];

                l--;

                j--;

            }

        }

    }

    printf("the number of duplicate elements in the array = %d\n",c);

    printf("the new array is:-\n");

*for*(int i=0;i<l;i++)

    printf("%d",arr[i]);

    int temp;

*for*(int i=0;i<l/2;i++)

    {

        temp=arr[i];

        arr[i]=arr[l-i-1];

        arr[l-i-1]=temp;

    }

    printf("\nthe new reversed array is:-\n");

*for*(int i=0;i<l;i++)

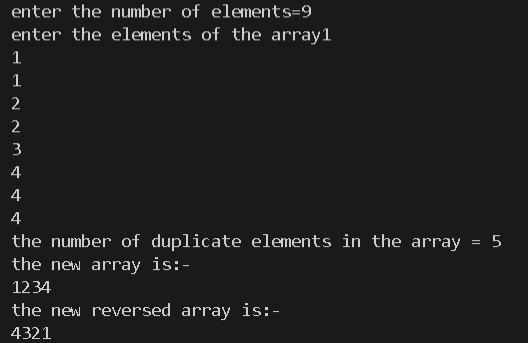
    printf("%d",arr[i]);

    free(arr);

*return* 0;

}

Output:



1. Write a menu-driven program for accepting values in two square matrices of 3x3 dimension and generate their sum, difference and product.

Sum can be done by running nested loops for rows and columns respectively and summing the respective elements and storing it in a new matrix.

Same applies for diiference except now the difference is stored in the new array.

For product we multiply respective elements of row from 1st matrix and column of 2nd matrix then sum of products is stored in the new matrix.

Program:

*#include*<stdio.h>

*#include*<stdlib.h>

*#include*<math.h>

int main()

{

    int a[10][10];

    int b[10][10];

    int n,choice;

    int c[10][10];

    printf("enter the number of rows/columns of 1st & 2nd array=");

    scanf("%d",&n);

    printf("enter the 1st array=");

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

    {

*for*(int j=0;j<n;j++)

        {

            scanf("%d",&a[i][j]);

        }

    }

    printf("enter the 2nd array=");

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

    {

*for*(int j=0;j<n;j++)

        {

            scanf("%d",&b[i][j]);

        }

    }

    printf("1 for sum , 2 for difference and 3 for product");

    printf("enter the choice=");

    scanf("%d",&choice);

*switch*(choice)

    {

*case* 1:

        printf("the reslutant array is:-\n");

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

        {

*for*(int j=0;j<n;j++)

            {

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

            }

        }

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

        {

*for*(int j=0;j<n;j++)

            {

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

            }

            printf("\n");

        }

*break*;

*case* 2:

        printf("the reslutant array is:-\n");

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

        {

*for*(int j=0;j<n;j++)

            {

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

            }

        }

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

        {

*for*(int j=0;j<n;j++)

            {

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

            }

            printf("\n");

        }

*break*;

*case* 3:

        printf("the reslutant array is:-\n");

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

        {

*for*(int j=0;j<n;j++)

            {

                int sum=0;

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

                {

                    sum+=a[i][c]\*b[c][j];

                }

                c[i][j]=sum;

            }

        }

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

        {

*for*(int j=0;j<n;j++)

            {

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

            }

            printf("\n");

        }

*break*;

*default*:

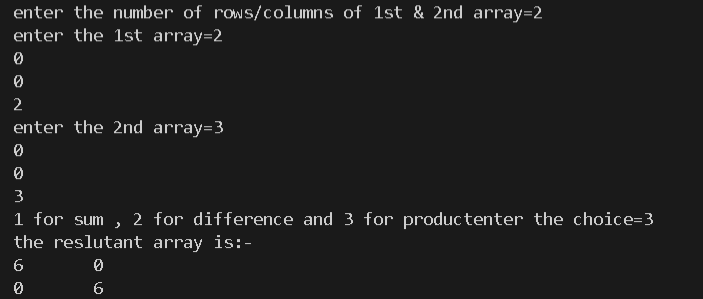
        printf("invalid choice");

    }

*return* 0;

}

Output:



1. Write a program to find the range of a set of integers entered by the user. Range is the difference between the smallest and biggest number in the list.

Take integers in an array find maximum and minimum then find ther difference.

Program:

*#include*<stdio.h>

*#include*<stdlib.h>

int main()

{

    int l,max=0,min;

    int\* arr;

    printf("enter the length of the array=");

    scanf("%d",&l);

    arr=(int\*)calloc(l,sizeof(int));

    printf("enter the elements of the array:-\n");

*for*(int i=0;i<l;i++)

    scanf("%d",&arr[i]);

    min=arr[0];

*for*(int i=0;i<l;i++)

    {

*if*(max<arr[i])

        max=arr[i];

*if*(min>arr[i])

        min=arr[i];

    }

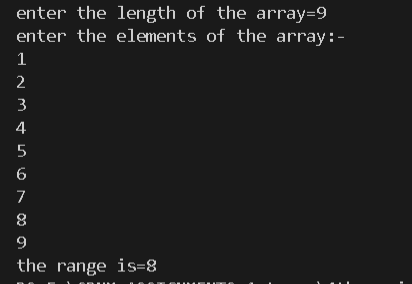
    printf("the range is=%d",max-min);

    free(arr);

*return* 0;

}

Output:



1. Write a C program which accepts ten integers from the user and prints them in ascending order. Use an array to store the integers.

Here I took 10 numbers from user and applied selection sort to sort them which has a time complexity of O(n\*n) for worst case.

Program:

*#include*<stdio.h>

*#include*<math.h>

*#include*<stdlib.h>

int main()

{

    int \*arr;

    int l;

    printf("enter the length of the array=");

    scanf("%d",&l);

    arr=(int\*)calloc(l,sizeof(int));

    printf("enter the elements of the array:-\n");

*for*(int i=0;i<l;i++)

    scanf("%d",&arr[i]);

    int temp;

*for*(int i=0;i<l-1;i++)

    {

*for*(int i=0;i<l-1;i++)

        {

*if*(arr[i]>arr[i+1])

            {

                temp=arr[i+1];

                arr[i+1]=arr[i];

                arr[i]=temp;

            }

        }

    }

    printf("the new array is:-\n");

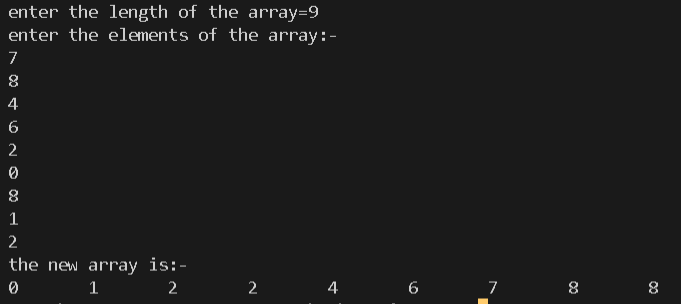
*for*(int i=0;i<l;i++)

    printf("%d\t",arr[i]);

*return* 0;

}

Output:



1. Write a C program which accepts roll numbers of ten students and marks obtained by them in five subjects and prints the names of the students who have obtained highest and second highest marks subject wise.

Take marks of students in five arrays for five subjects and name of the students in a 2d array then find highest and second highest for each array .

Program:

*#include*<stdio.h>

*#include*<string.h>

int main()

{

    char name[10][15];

    int roll[10];

    int a[10];

    int b[10];

    int c[10];

    int d[10];

    int e[10];

*for*(int i=0;i<10;i++)

    {

        printf("enter credentials of student %d:= i.e, name roll no and marks in 5 subjects\n",i+1);

        gets(name[i]);

        fflush(stdin);

        scanf("%d",&roll[i]);

        fflush(stdin);

        scanf("%d",&a[i]);

        fflush(stdin);

        scanf("%d",&b[i]);

        fflush(stdin);

        scanf("%d",&c[i]);

        fflush(stdin);

        scanf("%d",&d[i]);

        fflush(stdin);

        scanf("%d",&e[i]);

        fflush(stdin);

    }

    int n1,n2,max1=0,max2=0;

*for*(int i=0;i<10;i++)

    {

*if*(max1<a[i])

        {

            max1=a[i];

            n1=i;

        }

    }

    printf("the student who got highest marks in subject A is %s roll no-%d marks -%d\n",name[n1],roll[n1],max1);

*for*(int i=0;i<10;i++)

    {

*if*((a[i]>max2)&&(a[i]<max1))

        {

            max2=a[i];

            n2=i;

        }

    }

    printf("the student who got 2nd highest marks in subject A is %s roll no-%d marks -%d\n",name[n2],roll[n2],max2);

    max1=0;

    max2=0;

*for*(int i=0;i<10;i++)

    {

*if*(max1<b[i])

        {

            max1=b[i];

            n1=i;

        }

    }

    printf("the student who got highest marks in subject B is %s roll no-%d marks -%d\n",name[n1],roll[n1],max1);

*for*(int i=0;i<10;i++)

    {

*if*((b[i]>max2)&&(b[i]<max1))

        {

            max2=b[i];

            n2=i;

        }

    }

    printf("the student who got 2nd highest marks in subject B is %s roll no-%d marks -%d\n",name[n2],roll[n2],max2);

    max1=0;

    max2=0;

*for*(int i=0;i<10;i++)

    {

*if*(max1<c[i])

        {

            max1=c[i];

            n1=i;

        }

    }

    printf("the student who got highest marks in subject C is %s roll no-%d marks -%d\n",name[n1],roll[n1],max1);

*for*(int i=0;i<10;i++)

    {

*if*((c[i]>max2)&&(c[i]<max1))

        {

            max2=c[i];

            n2=i;

        }

    }

    printf("the student who got 2nd highest marks in subject C is %s roll no-%d marks -%d\n",name[n2],roll[n2],max2);

    max1=0;

    max2=0;

*for*(int i=0;i<10;i++)

    {

*if*(max1<d[i])

        {

            max1=d[i];

            n1=i;

        }

    }

    printf("the student who got highest marks in subject D is %s roll no-%d marks -%d\n",name[n1],roll[n1],max1);

*for*(int i=0;i<10;i++)

    {

*if*((d[i]>max2)&&(d[i]<max1))

        {

            max2=d[i];

            n2=i;

        }

    }

    printf("the student who got 2nd highest marks in subject D is %s roll no-%d marks -%d\n",name[n2],roll[n2],max2);

    max1=0;

    max2=0;

*for*(int i=0;i<10;i++)

    {

*if*(max1<e[i])

        {

            max1=e[i];

            n1=i;

        }

    }

    printf("the student who got highest marks in subject E is %s roll no-%d marks -%d\n",name[n1],roll[n1],max1);

*for*(int i=0;i<10;i++)

    {

*if*((e[i]>max2)&&(e[i]<max1))

        {

            max2=e[i];

            n2=i;

        }

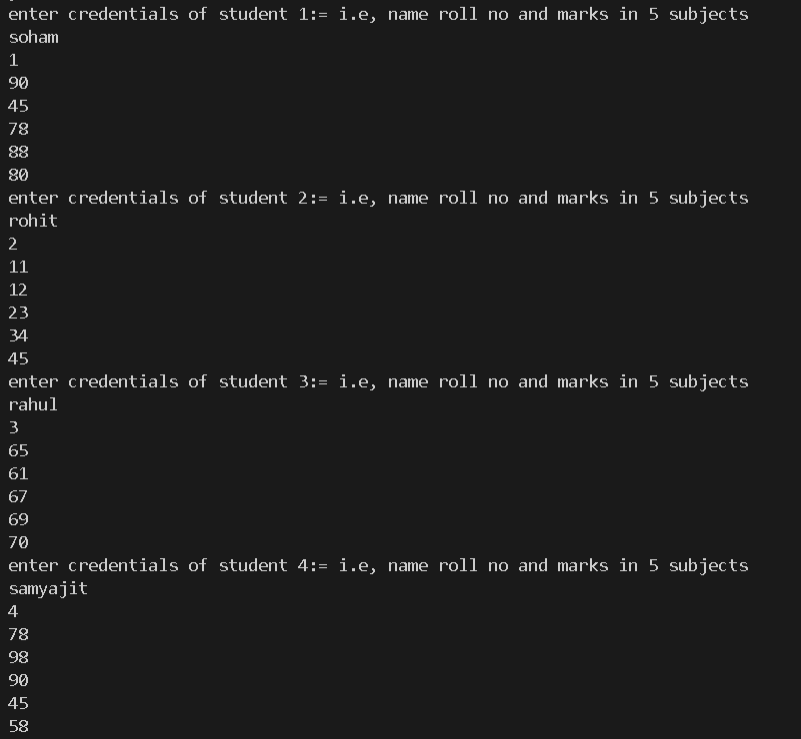
    }

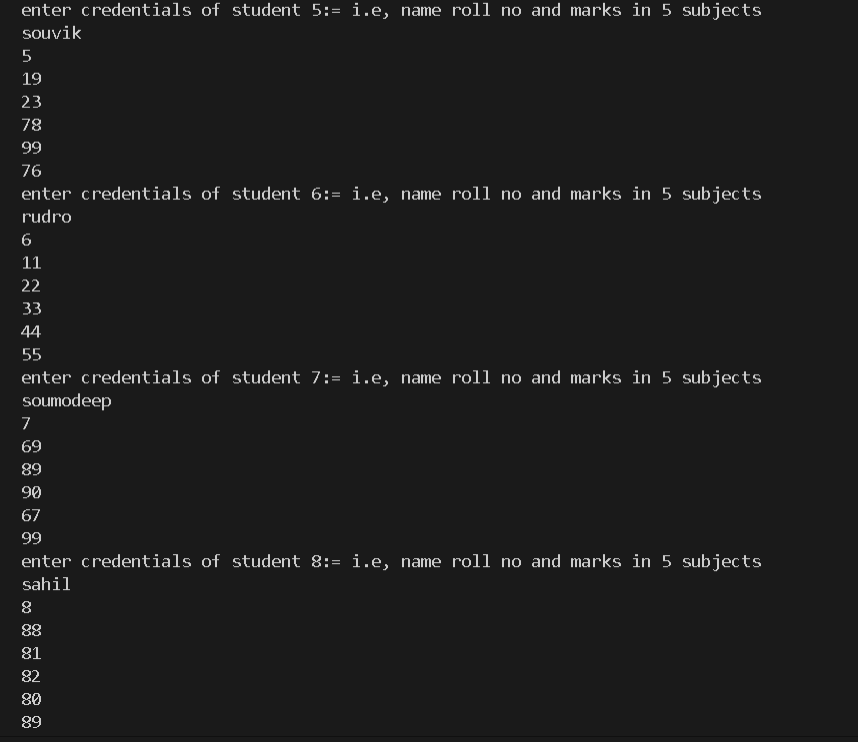
    printf("the student who got 2nd highest marks in subject E is %s roll no-%d marks -%d\n",(name[n2]),roll[n2],max2);

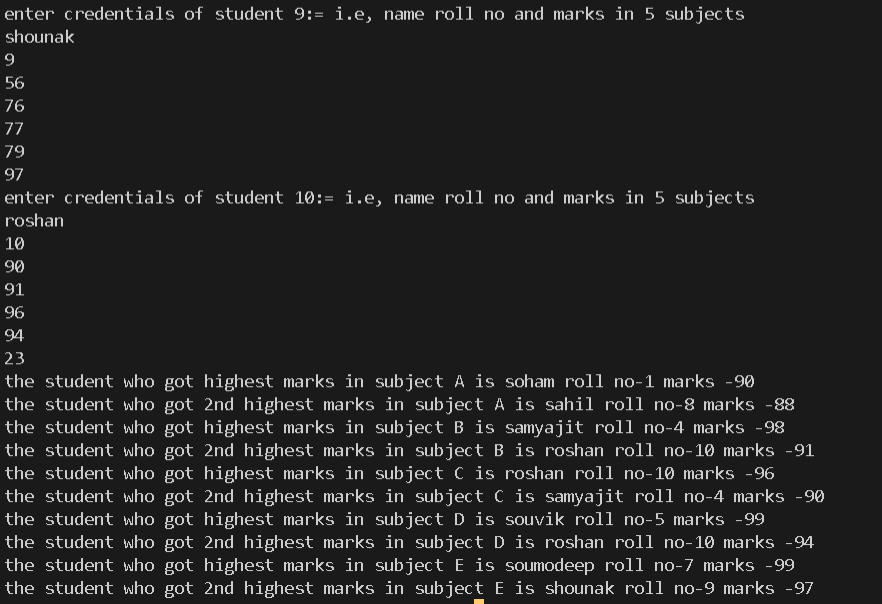
*return* 0;

}

Output:







1. Write a C program which accepts a matrix and prints its transpose.

Take matrix from user make another matrix and put ith row jth column of first matrix to jth row ith column of the new matrix.

Program:

*#include*<stdio.h>

*#include*<math.h>

*#include*<stdlib.h>

int main()

{

    int arr[10][10];

    int arr1[10][10];

    int n,m;

    printf("enter the rows & columns of the array \n");

    scanf("%d%d",&n,&m);

    printf("enter the elements of the array:-\n");

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

    {

*for*(int j=0;j<m;j++)

        scanf("%d",&arr[i][j]);

    }

    printf("the given array is:-\n");

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

    {

*for*(int j=0;j<m;j++)

        printf("%d\t",arr[i][j]);

        printf("\n");

    }

*for*(int j=0;j<m;j++)

    {

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

        {

            arr1[i][j]=arr[i][j];

        }

    }

    printf("the new array transposed is:-\n");

*for*(int j=0;j<m;j++)

    {

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

        printf("%d\t",arr1[i][j]);

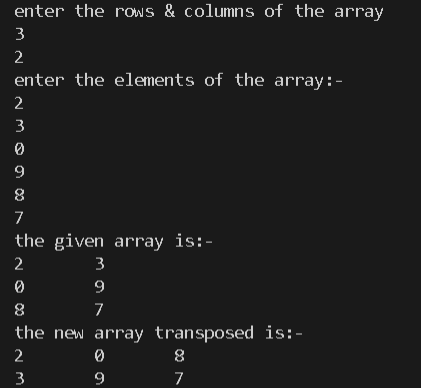
        printf("\n");

    }

*return* 0;

}

Output:



1. Write a C program to replace a square matrix by its transpose without using a second matrix

Take the initial matrix from the user and swap ith row jth column of first matrix to jth row ith column of the same matrix

Program:

*#include*<stdio.h>

*#include*<math.h>

int main()

{

    int arr[10][10];

    int n,temp;

    printf("enter the length of the square matrix=");

    scanf("%d",&n);

    printf("enter the matrix:-\n");

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

    {

*for*(int j=0;j<n;j++)

        {

            scanf("%d",&arr[i][j]);

        }

    }

    printf("the old matrix is:-\n");

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

    {

*for*(int j=0;j<n;j++)

        {

            printf("%d\t",arr[i][j]);

        }

        printf("\n");

    }

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

    {

*for*(int j=0;j<n;j++)

        {

*if*(j>=i)

            {

                temp=arr[i][j];

                arr[i][j]=arr[j][i];

                arr[j][i]=temp;

            }

        }

    }

    printf("the new transposed matrix is matrix is:-\n");

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

    {

*for*(int j=0;j<n;j++)

        {

            printf("%d\t",arr[i][j]);

        }

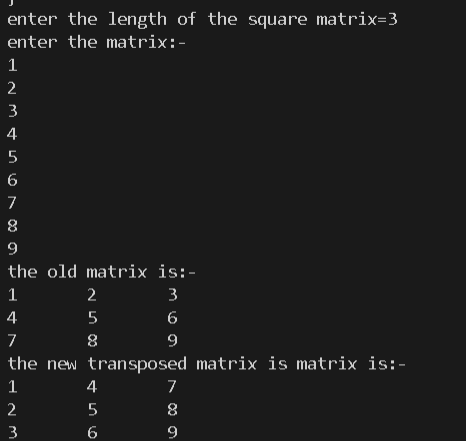
        printf("\n");

    }

*return* 0;

}

Output:



1. Consider the following procedure: i. Take as input any four-digit number, using at least two different digits. (Leading zeros are allowed.) ii. Arrange the digits in descending and then in ascending order to get two four-digit numbers, adding leading zeros if necessary. iii. Subtract the smaller number from the bigger number. Let the difference be the new four digit number. iv. Go back to step ii. The above process, known as Kaprekar's routine, will always reach a fixed point (Known as Kaprekar Constant). Write a C-Code to implement the algorithm given above and find out the constant number. Also create an output file ‘output.dat’ in the working folder and write the following with appropriate format for each step of iteration: The 4-digit number, the larger number, the smaller number and the difference of the larger and the smaller number. Note: A. The fixed point is achieved when in two consecutive steps the same number is obtained B. In C the binary arithmetic operation m%n gives the remainder when m is divided by n

Took the number from the user sorted it in ascending and descending order using selection sorting and reversing respectively,found the difference between ascending no and descending number stored it in an array and went on till the current element of the array is equal to the previous element of the array.

Program:

*#include*<stdio.h>

*#include*<math.h>

int main()

{

    int a[4],b[4],diff[100];

    int k=0,f=0;

    printf("enter the number (digit wise)=");

*for*(int i=0;i<4;i++)

    scanf("%d",&a[i]);

*while*(1)

    {

        int temp,num1=0,num2=0;

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

        {

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

            {

*if*(a[i]>a[i+1])

                {

                    temp=a[i];

                    a[i]=a[i+1];

                    a[i+1]=temp;

                }

            }

        }

*for*(int i=0;i<4;i++)

        {

            b[i]=a[4-i-1];

        }

*for*(int i=0;i<4;i++)

        {

            num1=num1\*10+a[i];

        }

*for*(int i=0;i<4;i++)

        {

            num2=num2\*10+b[i];

        }

*if*(num2>num1)

        k=num2-num1;

*else*

        k=num1-num2;

        diff[f++]=k;

*if*((f>=2)&&(diff[f-1]==diff[f-2]))

        {

            k=diff[f-1];

*break*;

        }

        int c=0;

*for*(int i=k;i>0;i/=10)

        c++;

        int l=3;

*if*(c==4)

        {

*for*(int i=k;i>0;i/=10)

            {

                a[l--]=i%10;

            }

            l=3;

        }

*else* *if*(c<4)

        {

            int g=3;

*for*(int i=k;i>0;i=i/10)

            {

*if*((g<=l-c)&&(g>=0))

                {

                    a[g--]=0;

                }

*else*

                {

                    a[g--]=i%10;

                }

            }

        }

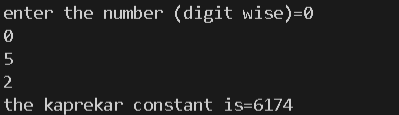
    }

    printf("the kaprekar constant is=%d",k);

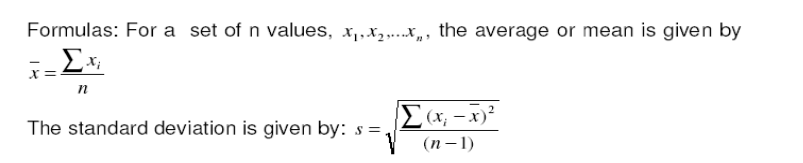
*return* 0;

}

Output:



1. Write a program which takes some numbers and computes the standard deviation of them.



Found the mean by traversing the user defined array and getting the sum of each element of the array by traversal and devided it with length to find mean.

Then standard devitation by again traversing the array and applieying the formula given above in the form of a mathematical expression.

Program:

*#include*<stdio.h>

*#include*<math.h>

*#include*<stdlib.h>

int main()

{

    int \* arr;

    int l;

    double mean,sum=0,sum1=0,s;

    printf("enter the length of the array=");

    scanf("%d",&l);

    arr=(int\*)malloc(l\*sizeof(int));

*for*(int i=0;i<l;i++)

    scanf("%d",&arr[i]);

*for*(int i=0;i<l;i++)

    sum+=arr[i];

    mean=sum/l;

    printf("the elements of the array is:-\n");

*for*(int i=0;i<l;i++)

    printf("%d\t",arr[i]);

    printf("\nmean=%f",mean);

*for*(int i=0;i<l;i++)

    {

        sum1+=pow(arr[i]-mean,2);

    }

    s=sqrt(sum1/(l-1));

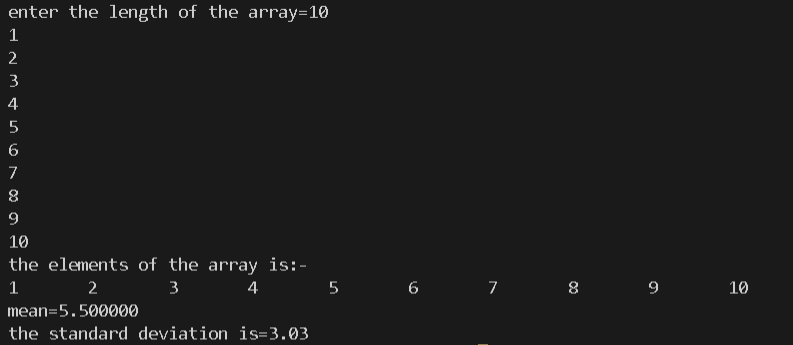
    printf("\nthe standard deviation is=%0.2f",s);

    free(arr);

*return* 0;

}

Output:



# ASSIGNMENT 5

1. Write recursive functions for following tasks. a. Binary equivalent of a number. b. Sum of individual digits of a number passed as argument.

For converting decimal to binary I stored the remainder in n%2,and sent n/2 for next recursion till n becomes 1.

For storing the sum of digit I extracted the digits by n%10 and made another function call n/10 tilln becomes 0.

Program:

*#include*<stdio.h>

*#include*<math.h>

int dec\_to\_bin(int n)

{

*if*(n==1)

*return* 1;

*else*

*return*(n%2+dec\_to\_bin(n/2)\*10);

}

int sum\_of\_digits(int n)

{

*if*(n==0)

*return* 0;

*else*

*return* (n%10+sum\_of\_digits(n/10));

}

int main ()

{

    int n;

    char choice;

    printf("enter the number=");

    scanf("%d",&n);

    fflush(stdin);

    printf("enter the choice=");

    scanf("%c",&choice);

    fflush(stdin);

*switch* (choice)

    {

*case* 'a':

    printf("the binary equivalent of the number is=%d",dec\_to\_bin(n));

*break*;

*case* 'b':

    printf("the sum of digits is %d",sum\_of\_digits(n));

*break*;

*default*:

    printf("invalid choice");

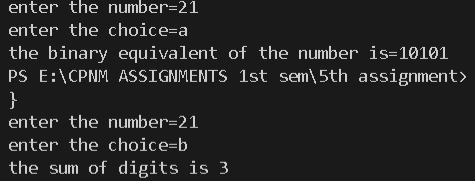
*break*;

    }

*return* 0;

}

Output:



1. Write a C program using functions which accepts a string from the user and performs the following tasks. a. Counts the number of characters in the string without using string library functions. b. Prints the reverse of the string without using string library functions.

Here I counted the number of characters by comparing each character of the character array with the space and If it is not a space then it has to be a character, for reverse I printed the string in reverse

Program:

*#include*<stdio.h>

*#include*<math.h>

int main()

{

    char str[100],choice;

    printf("enter the string:\n");

    gets(str);

    int i=0,c=0,j=0;

    printf("enter the choice=");

    scanf("%c",&choice);

*switch*(choice)

    {

*case* 'a':

*while*(str[i]!='\0')

        {

*if*(str[i]!=' ')

            c++;

            i++;

        }

        printf("the total no of characters in the given string is=%d",c);

*break*;

*case* 'b':

*while*(1)

        {

*if*(str[j]=='\0')

            {

*break*;

            }

*else*

            j++;

        }

        printf("the reversed string is:");

*while*(j>=0)

        {

            printf("%c",str[j]);

            j--;

        }

*break*;

*default*:

        printf("invalid choice");

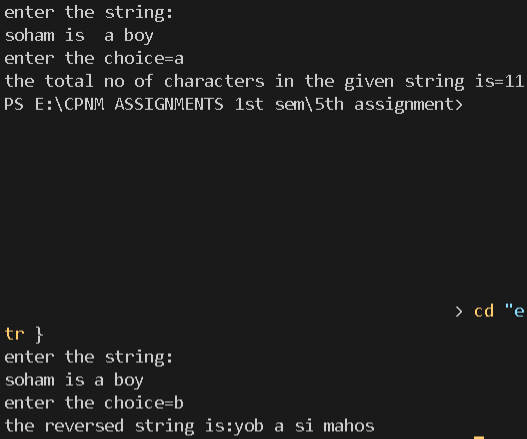
*break*;

    }

*return* 0;

}

Output:



1. Write a C program which accepts a full name from the user prints the initials. Eg. SRT for Sachin Ramesh Tendulkar.

Took the string as input from the user then added a space at the beginning of the string and if a space occurs before any character just print that.

Program:

*#include*<stdio.h>

*#include*<string.h>

int main()

{

    char name[30];

    printf("enter the string:-\n");

    gets(name);

    int l=strlen(name);

*for*(int i=l-1;i>=0;i--)

    {

        name[i+1]=name[i];

    }

    name[0]=' ';

    printf("the initials are as follows:");

*for*(int i=0;name[i]!='\0';i++)

    {

*if*((name[i]==' ')&&(name[i+1]!=' '))

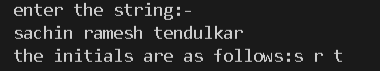
        printf("%c ",name[i+1]);

    }

*return* 0;

}

Output:



1. Write a program to count the number of occurrences of any two vowels in succession in a line of text.

Took consecutive elements of the character array using loops and if ith and i+1th elements were vowels then I increased the counter by 1.

Program:

*#include*<stdio.h>

*#include*<string.h>

int main()

{

    char str[30];

    int c=0;

    printf("enter the string=");

    gets(str);

*for*(int i=0;str[i]!='\0';i++)

    {

*if*(((str[i]=='a')||(str[i]=='A')||(str[i]=='e')||(str[i]=='E')||(str[i]=='i')||(str[i]=='I')||(str[i]=='o')||(str[i]=='O')||(str[i]=='u')||(str[i]=='U'))&&((str[i+1]=='a')||(str[i+1]=='A')||(str[i+1]=='e')||(str[i+1]=='E')||(str[i+1]=='i')||(str[i+1]=='I')||(str[i+1]=='o')||(str[i+1]=='O')||(str[i+1]=='u')||(str[i+1]=='U')))

        c++;

    }

    printf("the number of occurences of vowels in succession in a line is=%d",c);

*return* 0;

}

Output:



1. Write a program that converts (Do not use any string library function): a. A string like “123” to integer 123. b. An integer like 123 to string “123”.

For string to integer I used the atoi() function that is in the standard library of c. And for converting a user defined integer to string I used the concept of ASCII values.

Program:

*#include*<stdio.h>

*#include*<stdlib.h>

int main()

{

    char str[10];

    int num,c=0,sum=0;

    char choice,temp;

    printf("enter the choice=");

    scanf("%c",&choice);

    fflush(stdin);

*switch*(choice)

    {

*case* 'a':

        printf("enter the string=");

        gets(str);

        num=atoi(str);

        printf("%d",num);

*break*;

*case* 'b':

        printf("enter the number=");

        scanf("%d",&num);

*for*(int i=num;i>0;i/=10)

        {

            sum=sum\*10+i%10;

        }

*for*(int i=sum;i>0;i/=10)

        {

            str[c++]=(char)(i%10+48);

        }

        printf("the number in string is=");

*for*(int i=0;i<c;i++)

        printf("%c",str[i]);

*break*;

*default*:

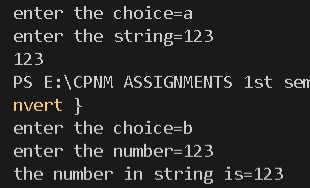
        printf("invalid choice");

    }

*return* 0;

}

Output:



1. Write a C program which accepts a string from the user and performs the following tasks. (Do not use any string library function. ) a. Check whether it is palindrome or not. [Example of a palindrome string: "abcba", "abba"] b. Counts the number of characters and words in it.

Traversed the string from beginning and end simultaeneously till the middle of the string and if each and every character matches then the string is palindrome.

Program:

*#include*<stdio.h>

int main()

{

    char str1[30];

    printf("enter the string=");

    gets(str1);

    char choice;

    int f=0,c=0,w=0,k;

    printf("enter the choice=");

    scanf("%c",&choice);

*switch*(choice)

    {

*case* 'a':

*for*(int i=0;;i++)

        {

*if*(str1[i]=='\0')

            {

                k=i;

*break*;

            }

        }

*for*(int i=0;i<k/2;i++)

        {

*if*(str1[i]!=str1[k-i-1])

            {

                f=1;

*break*;

            }

        }

*if*(f==1)

        printf("the string is not pallindrome");

*else*

        printf("the string is pallindrome");

*break*;

*case* 'b':

*for*(int i=0;;i++)

        {

*if*(str1[i]=='\0')

            {

                k=i;

*break*;

            }

        }

*for*(int i=k;i>=0;i--)

        str1[k+1]=str1[k];

        str1[0]=' ';

*for*(int i=0;i<k+1;i++)

        {

*if*((str1[i]==' ')&&(str1[i+1]!=' '))

            w++;

*if*(str1[i]!=' ')

            c++;

        }

        printf("the number of words are %d and characters are %d in the given string",w,c);

*break*;

*default*:

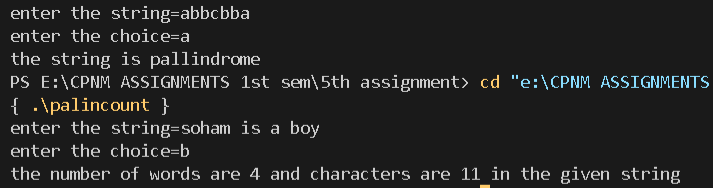
        printf("invalid choice");

    }

*return* 0;

}

Output:



1. Write a program in C to store n numbers in an array and print the elements using pointers. Also compute the sum of all elements of that array using pointers.

The array pointer points towards the first element of the array and with increase in value of pointer by 1 we can traverse the entire array and summed up individual elements .

Program:

*#include*<stdio.h>

int main()

{

    int arr[30];

    int \*ptr;

    int l,sum=0;

    printf("enter the length of the array=");

    scanf("%d",&l);

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

*for*(int i=0;i<l;i++)

    scanf("%d",&arr[i]);

    ptr=&arr[0];*//or ptr=&arr but error showing;*

*for*(int i=0;i<l;i++)

    {

        sum=sum+\*(ptr);

        ptr=ptr+1;

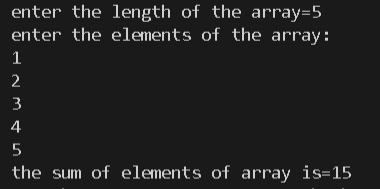
    }

    printf("the sum of elements of array is=%d",sum);

*return* 0;

}

Output:



8. Write a C function which accepts a string str1 and returns a new string str2 which is str1 with each word reversed. Do not use any string library function.

For reversing words I just reversed each words by swapping 1st and last element and 2nd and 2nd last element and so on till the middle of the word and traversed every word.

For reversing order of the words I took the words from the end of the array and put them to the otter end of another array.

Program:

*#include*<stdio.h>

*#include*<math.h>

char \* rev\_wrd1(char str[])

{

    static char res[50];

    char word[15],temp;

    int c=0;

    int l=0,d;

*for*(int i=0;;i++)

    {

*if*(str[i]=='\0')

        {

            d=i;

*break*;

        }

    }

    str[d]=' ';

    str[d+1]='\0';

*for*(int i=0;str[i]!='\0';i++)

    {

*if*(str[i]==' ')

        {

            word[c]='\0';

*for*(int j=0;j<c/2;j++)

            {

                temp=word[j];

                word[j]=word[c-1-j];

                word[c-1-j]=temp;

            }

*for*(int k=0;word[k]!='\0';k++)

            {

                res[l++]=word[k];

            }

            res[l++]=' ';

            c=0;

        }

*else*

        {

            word[c++]=str[i];

        }

    }

*return* (res);

}

char \* rev\_wrd2(char str[])

{

    static char res[50];

    int d,c=0;

    char word[15];

*for*(int i=0;;i++)

    {

*if*(str[i]=='\0')

        {

            d=i;

*break*;

        }

    }

    str[d]=' ';

    str[d+1]='\0';

    res[d--]='\0';

*for*(int j=0;str[j]!='\0';j++)

    {

*if*(str[j]==' ')

        {

            word[c]='\0';

            int l=c-1;

*for*(int f=l;f>=0;f--)

            {

*if*(d>=0)

                {

                    res[d--]=word[f];

                }

            }

            res[d--]=' ';

            c=0;

        }

*else*

        {

            word[c++]=str[j];

        }

    }

*return* (res);

}

int main()

{

    char str[50];

    char \* ptr;

    printf("enter the string=");

    gets(str);

    int choice;

    printf("enter the choice=");

    scanf("%d",&choice);

*if*(choice==1)

    ptr=rev\_wrd1(str);

*if* (choice==2)

    ptr=rev\_wrd2(str);

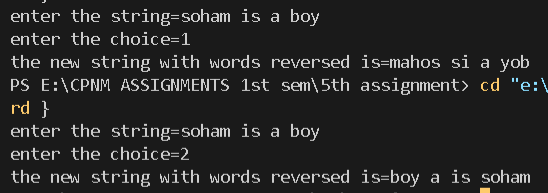
    printf("the new string with words reversed is=");

    puts(ptr);

*return* 0;

}

Output:



1. Write a function squeeze(s,c) which removes all occurrences of the character c from the strings.

As soon as we found that the character c is in a character array then just delete that element from that string.

Program:

*#include*<stdio.h>

*#include*<string.h>

char\* squeeze(char s[30],char c)

{

*for*(int i=0;s[i]!='\0';i++)

    {

*if*(s[i]==c)

        {

*for*(int j=i;s[j]!='\0';j++)

            {

                s[j]=s[j+1];

            }

            i--;

        }

    }

*return* s;

}

int main()

{

    char str[30];

    char \*ptr;

    char c;

    printf("enter the string=");

    gets(str);

    printf("enter the character to be squeezed=");

    scanf("%c",&c);

    ptr=squeeze(str,c);

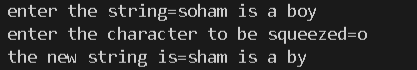
    printf("the new string is=");

    puts(ptr);

*return* 0;

}

Output:



1. Write the function strend(s,t), which returns 1 if the string t occurs at the end of the string s, and zero otherwise.

Take both strings from the user then check each and every element from the end of the string 1 till the end of length of second string.

Program:

*#include*<stdio.h>

*#include*<string.h>

int stend(char s[50],char t[30])

{

    int l=strlen(s);

    int k=strlen(t);

    int c=0;

*for*(int i=l-1,j=k-1;i>=0,j>=0;i--,j--)

    {

*if*(s[i]!=t[j])

        {

            c=1;

*break*;

        }

    }

*if*(c==0)

*return* 1;

*else*

*return* 0;

}

int main()

{

    char s[50];

    char t[30];

    printf("enter the string=");

    gets(s);

    fflush(stdin);

    printf("enter the string to be checked=");

    gets(t);

*if*(stend(s,t)==1)

    printf("the string %s occurs at the end of string %s",t,s);

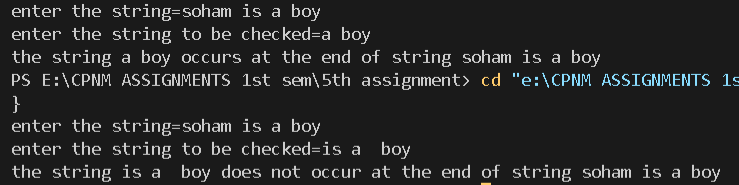
*else*

    printf("the string %s does not occur at the end of string %s",t,s);

*return* 0;

}

Output:



# ASSIGNMENT 6

1. Write a menu-driven program for finding roots of a nonlinear equation using Bisection, Regula Falsi and Newton-Raphson method.

I initially take the upper limit and lower limit from the user then perform the bisection or newton raphson method or the false position method of numerical methods.for newton raphson only one nearest root is required.

Program:

*#include*<stdio.h>

*#include*<math.h>

*#define* f(x) x\*x\*x-4\*x+1

*#define* df(x) 3\*x\*x-4

*#define* e 0.001

int main()

{

    float x0,x1,x2,f1,f0,f2,df0;

    int choice,i=0,n;

    printf("enter the choice=");

    scanf("%d",&choice);

*switch*(choice)

    {

*case* 1:

        printf("enter x0=");

        scanf("%f",&x0);

        fflush(stdin);

        printf("enter x1=");

        scanf("%f",&x1);

*if*(f(x0)\*f(x1)>0)

        {

            printf("invalid inputs");

        }

*else*

        {

*do*

            {

                f0=f(x0);

                f1=f(x1);

                x2=(x0+x1)/2;

                f2=f(x2);

*if*(f0\*f2<0)

                {

                    x1=x2;

                }

*else*

                {

                    x0=x2;

                }

                i++;

                } *while* (fabs(f2)>e);

                printf("root=%f",x2);

                printf("\nnumber of iterrations=%d",i);

        }

*break*;

*case* 2:

        printf("enter x0=");

        scanf("%f",&x0);

        fflush(stdin);

        printf("enter x1=");

        scanf("%f",&x1);

*if*(f(x0)\*f(x1)>0)

        {

            printf("invalid inputs");

        }

*else*

        {

*do*

            {

                f0=f(x0);

                f1=f(x1);

                x2=((x0\*f1)-(x1\*f0))/(f1-f0);

                f2=f(x2);

*if*(f0\*f2<0)

                {

                    x1=x2;

                }

*else*

                {

                    x0=x2;

                }

                i++;

            } *while* (fabs(f2)>e);

            printf("root=%f",x2);

            printf("\nnumber of iterrations=%d",i);

        }

*break*;

*case* 3:

        printf("enter x0=");

        scanf("%f",&x0);

        printf("enter the maximum number of iterations=");

        scanf("%d",&n);

*do*

        {

            f0=f(x0);

            df0=df(x0);

*if*(df0==0.0)

            {

                printf("math error");

*break*;

            }

*if*(i<n)

            {

                x1=x0-f0/df0;

                i++;

                x0=x1;

                f1=f(x1);

            }

        } *while* (fabs(f1)>e);

        printf("root=%f",x1);

        printf("\nnumber of iterrations=%d",i);

*break*;

*default*:

        printf("invalid choice");

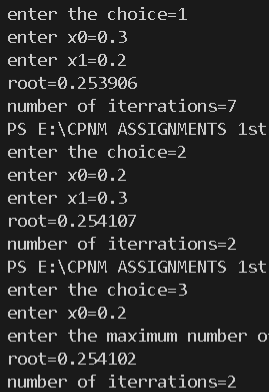
*break*;

    }

*return* 0;

}

Output:



1. Use the above program to find 3 roots of the equation xtan(x)=c where c is a user-input constant. Use both the bisection method and Newton-Raphson method.

The equation to be solved is xtan(x)-c=0,then appy the newton raphson and bisection method using these we find three roots for this equation by getting three ranges or nearest roots from the user.

Program:

*#include*<stdio.h>

*#include*<stdlib.h>

*#include* <math.h>

*#include*<limits.h>

*#define* e 0.001

double f(double x,double c)

{

*return* x\*tan(x)-c;

}

double df(double x)

{

*return* tan(x)+x\*(1/(cos(x)\*cos(x)));

}

void bisection\_method(double x0,double x1,double c)

{

    double f0,f1,f2,x2,a,b;

    a=x0;

    b=x1;

    int k=0;

    printf("\n  f(x0)\t         f(x1)\t      f(x2)\n");

*do*

    {

        f0=f(a,c);

        f1=f(b,c);

*if*(f0\*f1>0)

        {

           printf("no root can be found between %lf & %lf",x0,x1);

           k=1;

        }

        x2=(a+b)/2;

        f2=f(x2,c);

        printf("\n%lf\t%lf\t%lf",f0,f1,f2);*//printing f1,f0.*

*if*(f0\*f2>0)

        {

            a=x2;

        }

*else*

        {

            b=x2;

        }

    } *while* (fabs(f2)>e);

*if*(k==1)

    printf("no root could be found between %lf and %lf",x0,x1);

*else*

    printf("\nthe root using newton ralphson method is=%lf\n",x2);

}

void newton\_ralphson\_method(double x0,double c,int n)

{

    double g,f0,df0,f1,df1;

    double x1;

    int i=0,k=0;

    g=x0;

    printf("\n  x1\t          f(x1)\t          df(x1)\n");

*do*

    {

        f0=f(g,c);

        df0=df(g);

*if*(df0==0)

        {

            printf("math error");

            k=1;

*break*;

        }

*if*(i<n)

        {

            x1=g-(f0/df0);

            i++;

            f1=f(x1,c);

            df1=df(x1);

            printf("%lf\t%lf\t%lf\n",x1,f1,df1);*//printing x1,f1,df1*

            g=x1;

        }

    } *while* (fabs(f1)>e);

*if*(k==0)

    {

        printf("\nthe root using newton ralphson method =%lf\n",x1);

    }

*else*

    printf("\nthe root could not be found\n");

}

int main()

{

    double c;

    int n;

    printf("enter the value of c=");

    scanf("%lf",&c);

    fflush(stdin);

    double x0,x1,f0,f1;

    double root;

    int choice;

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

    {

        printf("\nenter the details of the root number %d:\n",i+1);

        printf("enter the choice:=\n1:bisection method\n2:newton ralphson method\n=");

        scanf("%d",&choice);

*if*(choice==1)

        {

            printf("enter the value of x0=");

            scanf("%lf",&x0);

            fflush(stdin);

            printf("enter the value of x1=");

            scanf("%lf",&x1);

            bisection\_method(x0,x1,c);

        }

*else* *if*(choice==2)

        {

            printf("enter the value of x0=");

            scanf("%lf",&x0);

            fflush(stdin);

            printf("enter the maximum number of iterations=");

            scanf("%d",&n);

            newton\_ralphson\_method(x0,c,n);

        }

*else*

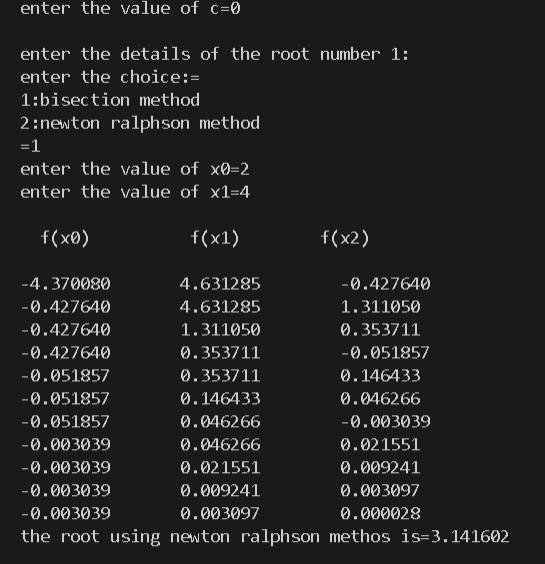
            printf("invalid choice");

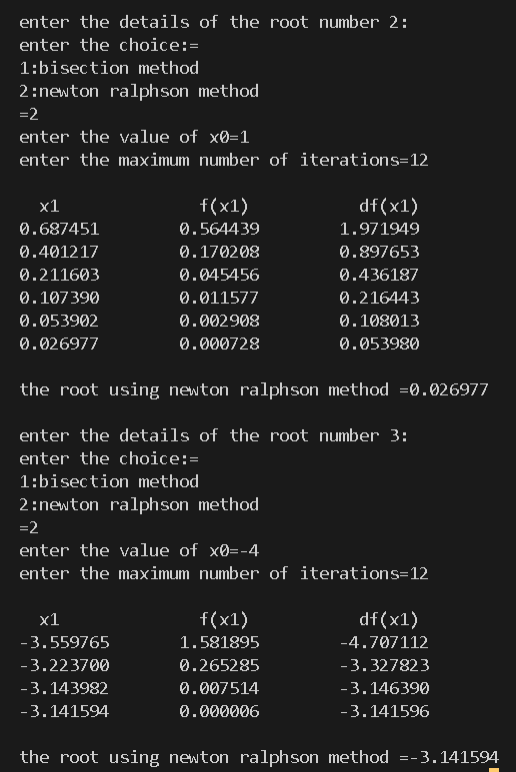
    }

*return* 0;

}

Output:





1. There are three real roots of the equation x 3 – 2.5x 2 – 2.46x + 3.96 = 0 in the domain [-4, +4]. Write a program to first find out the disjoint subintervals in the given domain those cover the roots. Hence find the roots by Newton-Raphson method.

Start x0=-4 and x1=-3 and if a root exists between them then we apply newton raphson method else increase x1/upper limit by 0.01 .

Program:

*#include*<stdio.h>

*#include*<math.h>

*#include*<stdlib.h>

*#define* f(x) x\*x\*x-2.5\*x\*x-2.46\*x+3.96

*#define* df(x) 3\*x\*x-2.5\*2\*x-2.46

*#define* e 0.001

void newton\_ralphson(float x0,int n,float r1,float r2)

{

    float f0,df0,f1,df1,x1,a;

    int i=0;

    a=x0;

*do*

    {

       f0=f(a);

       df0=df(a);

*if*(df0==0)

       {

           printf("mathematical error");

*break*;

       }

*if*(i<n)

       {

           x1=a-f0/df0;

           i++;

           a=x1;

           f1=f(a);

           df1=df(a);

       }

    }*while*(fabs(f1)>e);

    printf("the root using newton ralphson method within [%f,%f)is=%0.3f\n\n\n",r1,r2,x1);

}

int main()

{

    float x0,x1,root1,root2,root3;

    int n,no\_of\_root=0;

    float f0,f1;

    x0=-4.0;

    x1=-3.0;

    printf("enter the number of iterations(maximum)=");

    scanf("%d",&n);

*while*(no\_of\_root<3)

    {

        f0=f(x0);

        f1=f(x1);

*if*(f0\*f1<0)

        {

            printf("the root %d exists between [%f,%f)\n",no\_of\_root+1,x0,x1);

            newton\_ralphson(x1,n,x0,x1);

            no\_of\_root++;

            x0=x1;

*if*(x1<4)

            x1=x1+1;

*if*(no\_of\_root==2)

            x1=4.0;

        }

*else*

        {

            x1=x1+0.01;

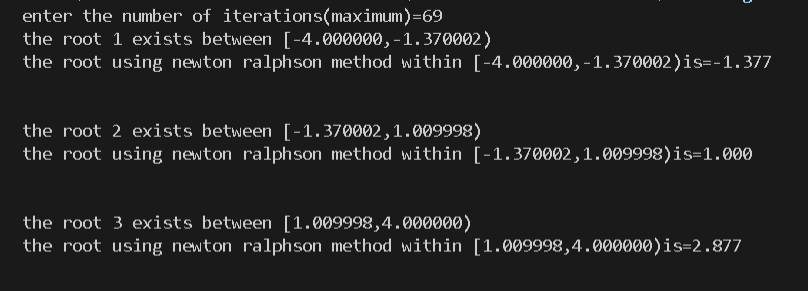
        }

    }

*return* 0;

}

Output:



# ASSIGNMENT 7