7. Armstrong number

#include<stdio.h>

int main()

{

int n,r,sum=0,temp;

printf("enter the number=");

scanf("%d",&n);

temp=n;

while(n>0)

{

r=n%10;

sum=sum+(r\*r\*r);

n=n/10;

}

if(temp==sum)

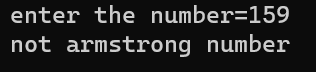
printf("armstrong number ");

else

printf("not armstrong number");

return 0;

}



8. To reverse a number

#include<stdio.h>

int main()

{

int n, reverse=0, rem;

printf("Enter a number: ");

scanf("%d", &n);

while(n!=0)

{

rem=n%10;

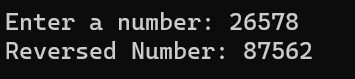
reverse=reverse\*10+rem;

n/=10;

}

printf("Reversed Number: %d",reverse);

return 0; }



9.Palindrom

#include<stdio.h>

int main()

{

int n,r,sum=0,temp;

printf("enter the number=");

scanf("%d",&n);

temp=n;

while(n>0)

{

r=n%10;

sum=(sum\*10)+r;

n=n/10;

}

if(temp==sum)

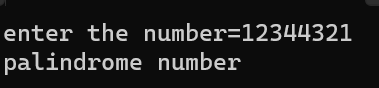
printf("palindrome number ");

else

printf("not palindrome");

return 0;

}



10.Prime or not

#include <stdio.h>

int main() {

int n, i, flag = 0;

printf("Enter a positive integer: ");

scanf("%d", &n);

// 0 and 1 are not prime numbers

// change flag to 1 for non-prime number

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

flag = 1;

for (i = 2; i <= n / 2; ++i) {

// if n is divisible by i, then n is not prime

// change flag to 1 for non-prime number

if (n % i == 0) {

flag = 1;

break;

}

}

// flag is 0 for prime numbers

if (flag == 0)

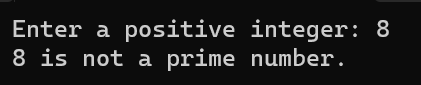
printf("%d is a prime number.", n);

else

printf("%d is not a prime number.", n);

return 0;

}



11.All factors of natural number

int main()

{

int i, num;

/\* Input number from user \*/

printf("Enter any number to find its factor: ");

scanf("%d", &num);

printf("All factors of %d are: \n", num);

/\* Iterate from 1 to num \*/

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

{

/\*

\* If num is exactly divisible by i

\* Then i is a factor of num

\*/

if(num % i == 0)

{

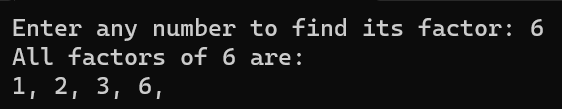
printf("%d, ",i);

}

}

return 0;

}



 12.bubble sort

13.To compute and print matrix multiplication

#include<stdio.h>

int main() {

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

printf("Enter the value of N (N <= 10): ");

scanf("%d", & n);

printf("Enter the elements of Matrix-A: \n");

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

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

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

}

}

printf("Enter the elements of Matrix-B: \n");

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

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

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

}

}

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

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

c[i][j] = 0;

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

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

}

}

}

printf("The product of the two matrices is: \n");

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

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

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

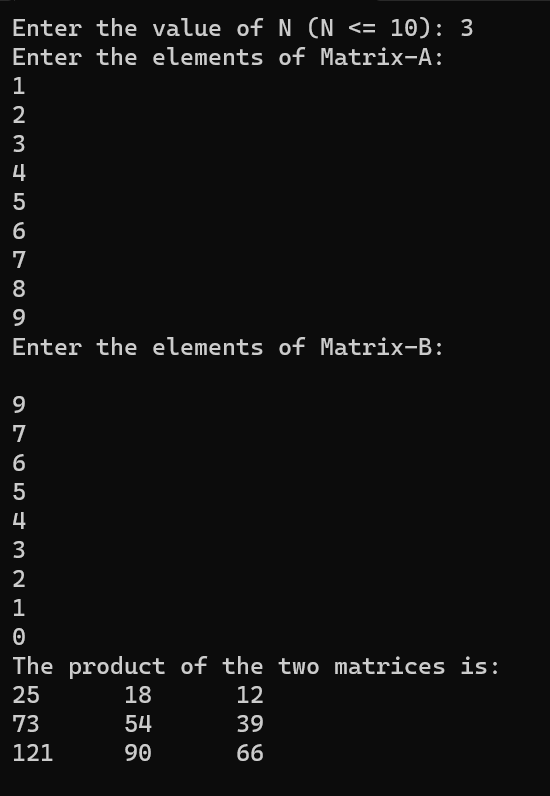
}

printf("\n");

}

return 0;

}



14.Binary search and linear search

#include<stdio.h>

int linearSearch(int arr[], int size, int element){

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

{

if(arr[i]==element){

return i;

}

}

return -1;

}

int binarySearch(int arr[], int size, int element){

int low, mid, high;

low = 0;

high = size-1;

// Keep searching until low <= high

while(low<=high){

mid = (low + high)/2;

if(arr[mid] == element){

return mid;

}

if(arr[mid]<element){

low = mid+1;

}

else{

high = mid -1;

}

}

return -1;

}

int main(){

// Unsorted array for linear search

// int arr[] = {1,3,5,56,4,3,23,5,4,54634,56,34};

// int size = sizeof(arr)/sizeof(int);

// Sorted array for binary search

int arr[] = {1,3,5,56,64,73,123,225,444};

int size = sizeof(arr)/sizeof(int);

int element = 444;

int searchIndex = binarySearch(arr, size, element);

printf("The element %d was found at index %d \n", element, searchIndex);

return 0;

}

