Q1. A positive integer is entered through the keyboard; write a recursive function to find the binary equivalent of the number.

Code:

//862041\_Naveen Kumar Tyagi\_Section F

#include<iostream>

using namespace std;

void binary\_form(int num,int bin[20],int i,int \*last\_index){

    //perform division and store remainder

    //and pass quotient to itself to do same again

    //until quotient became 0

    if(num>0){

        bin[i]=num%2;

        i++;

        binary\_form(num/2,bin,i,last\_index);

    }

    //when conversion is completed it store last index

    else if(num==0){

       \*last\_index=i-1;

    }

}

int main(){

    cout<<"862041\_Naveen Kumar Tyagi\_Section F";

    //last\_index is to store last index of bin array

    //upto which bits for num is stored

    int num,bin[20],last\_index;

    cout<<"\nEnter number: ";

    cin>>num;

    //call of function to convert decimal number to binary

    binary\_form(num,bin,0,&last\_index);

    cout<<"\nBinary equivalent: ";

    for(int i=last\_index;i>=0;i--){  //print binary equivalent

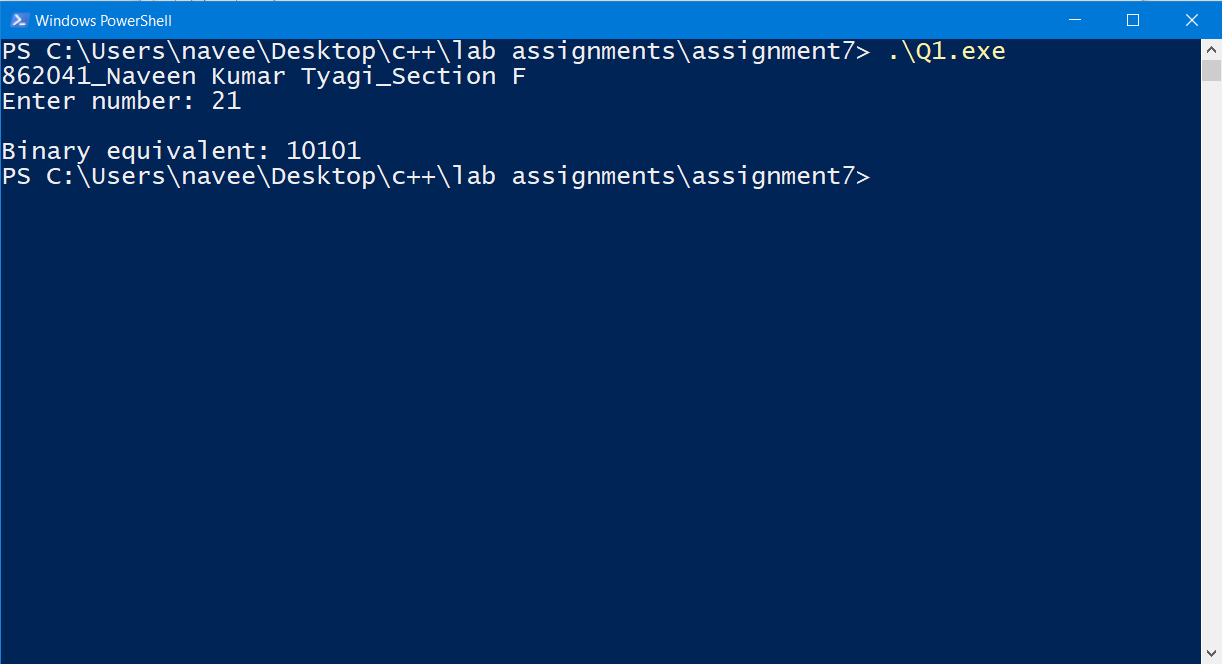
        cout<<bin[i];

    }

    return 0;

}

Output:



Q2. Write a recursive function that computes the sum of all numbers from 1 to n, where n is given as parameter.

Code:

//862041\_Naveen Kumar Tyagi\_Section F

#include<iostream>

using namespace std;

int sum=0;

//add number to sum and decrease it's value by 1

//then pass number to itself and do same

//until that number became 0

//returns final value of sum

int adder(int n){

    if(n>0){

        sum+=n--;

        adder(n);

    }

    return sum;

}

int main(){

    cout<<"862041\_Naveen Kumar Tyagi\_Section F";

    int N;

    cout<<"\nEnter N: ";

    cin>>N;

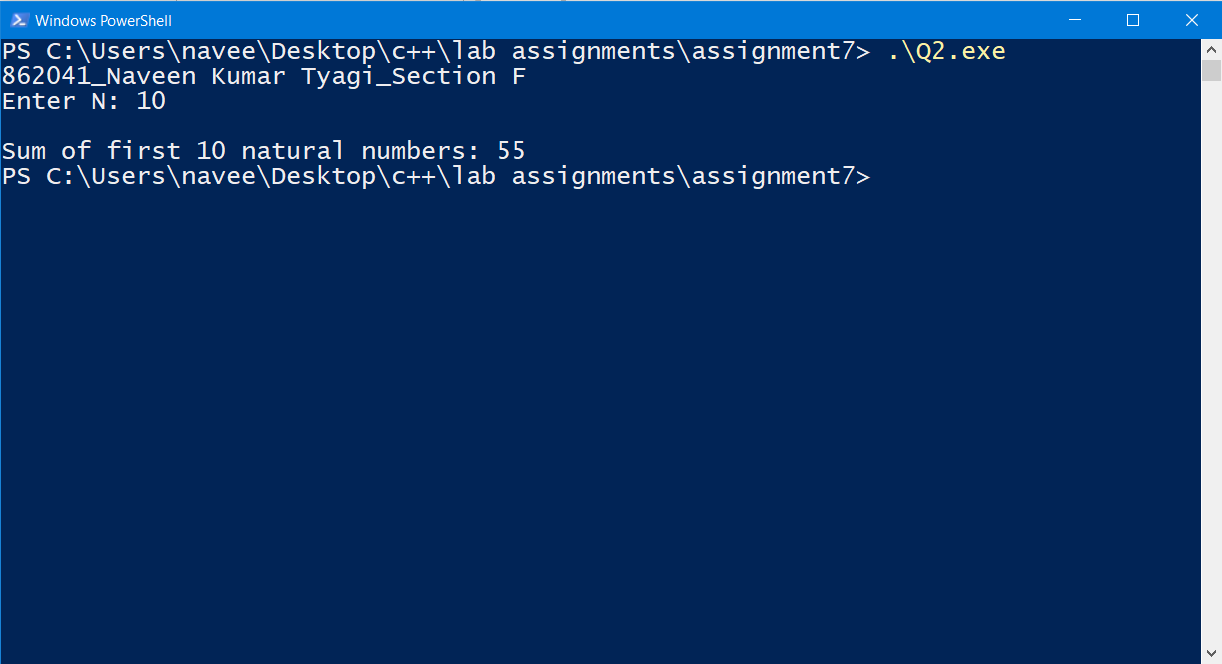
    //calling adder function to get sum

    cout<<"\nSum of first "<<N<<" natural numbers: "<<adder(N);

    return 0;

}

Output:



Q3. Write a program to print Fibonacci Series using recursion.

Code:

//862041\_Naveen Kumar Tyagi\_Section F

#include<iostream>

using namespace std;

int fibonacci\_num(int n){

    // first two numbers are 1

    if(n==1||n==2){

        return 1;

    }

    //call itself to get (n-1)th and (n-2)th value

    //num store the sum of that two values

    int num=fibonacci\_num(n-1)+fibonacci\_num(n-2);

    //return num(n th number) at last

    return num;

}

int main(){

    cout<<"862041\_Naveen Kumar Tyagi\_Section F";

    int N;

    cout<<"\nEnter N: ";

    cin>>N;

    cout<<"\nFirst "<<N<<" fibonacci numbers: ";

    for(int i=1;i<=N;i++){

        //calling function to return i th number

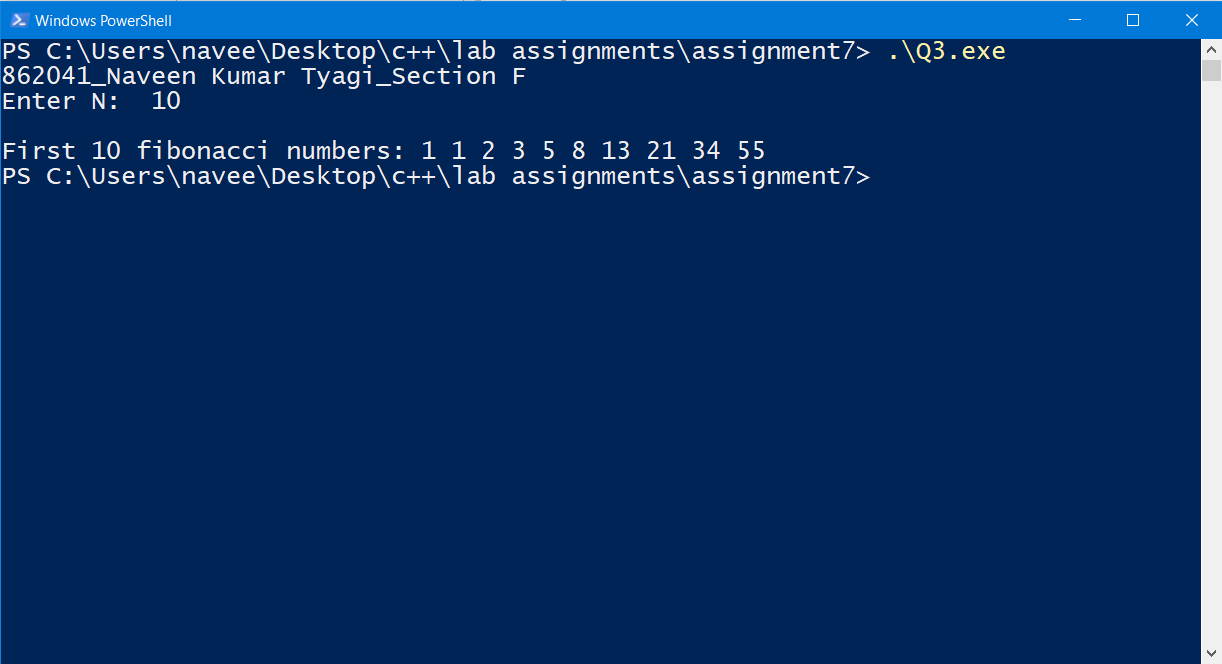
        cout<<fibonacci\_num(i)<<" ";

    }

    return 0;

}

Output:



Q4. Write a recursive function for towers of Hanoi problem

Code:

//862041\_Naveen Kumar Tyagi\_Section F

#include<iostream>

using namespace std;

void towerOfHanoi(int n\_disks,char src\_rod,char des\_rod,char aux\_rod) {

    //move last disk from source rod to destination rod

    if (n\_disks == 1){

        cout<<"Move disk 1 from rod "<<src\_rod<<" to rod "<<des\_rod<<endl;

        return;

    }

    //move n-1 disks  from source rod to auxillary rod

    //considering aux. rod to be destination rod for n-1 disks

    towerOfHanoi(n\_disks - 1,src\_rod,aux\_rod,des\_rod);

    cout<<"Move disk "<<n\_disks<<" from rod "<<src\_rod<<" to rod "<<des\_rod << endl;

    //move n-1 disks from auxillary rod to destination rod

    towerOfHanoi(n\_disks - 1,aux\_rod,des\_rod,src\_rod);

}

int main(){

    cout<<"862041\_Naveen Kumar Tyagi\_Section F";

    int n\_disks; // Number of disks

    cout<<"\nEnter number of disks: ";

    cin>>n\_disks;

    //A is source rod

    //B is auxillary rod

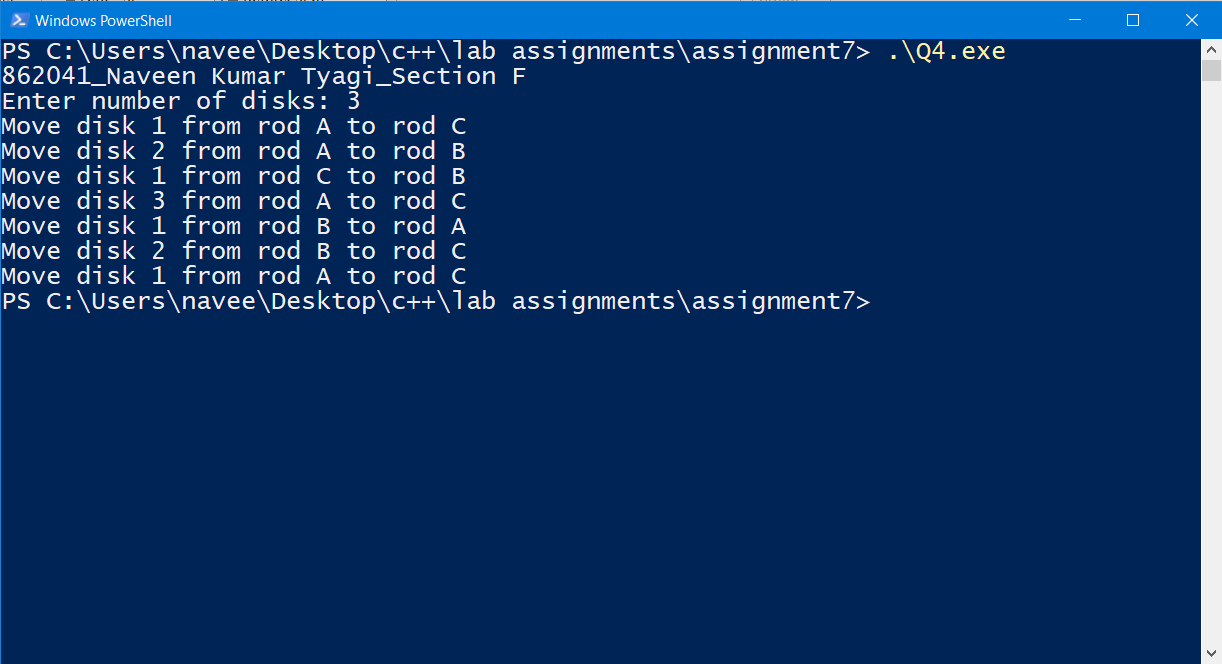
    //C is destination rod

    towerOfHanoi(n\_disks,'A','C','B'); // A, B and C are names of rods

    return 0;

}

Output:



Q5. Write a program to store n elements in an array and print the elements using pointer.

Code:

//862041\_Naveen Kumar Tyagi\_Section F

#include<iostream>

using namespace std;

int main(){

    cout<<"862041\_Naveen Kumar Tyagi\_Section F";

    int n;

    cout<<"\nEnter size of array: ";

    cin>>n;

    int arr[n];

    cout<<"\nEnter elements of array: ";

    //take input and store in array

    for(int i=0;i<n;i++){

        cin>>arr[i];

    }

    //printing array elements using pointers

    int \*p=arr;

    cout<<"Array elements: ";

    for(int i=0;i<n;i++){

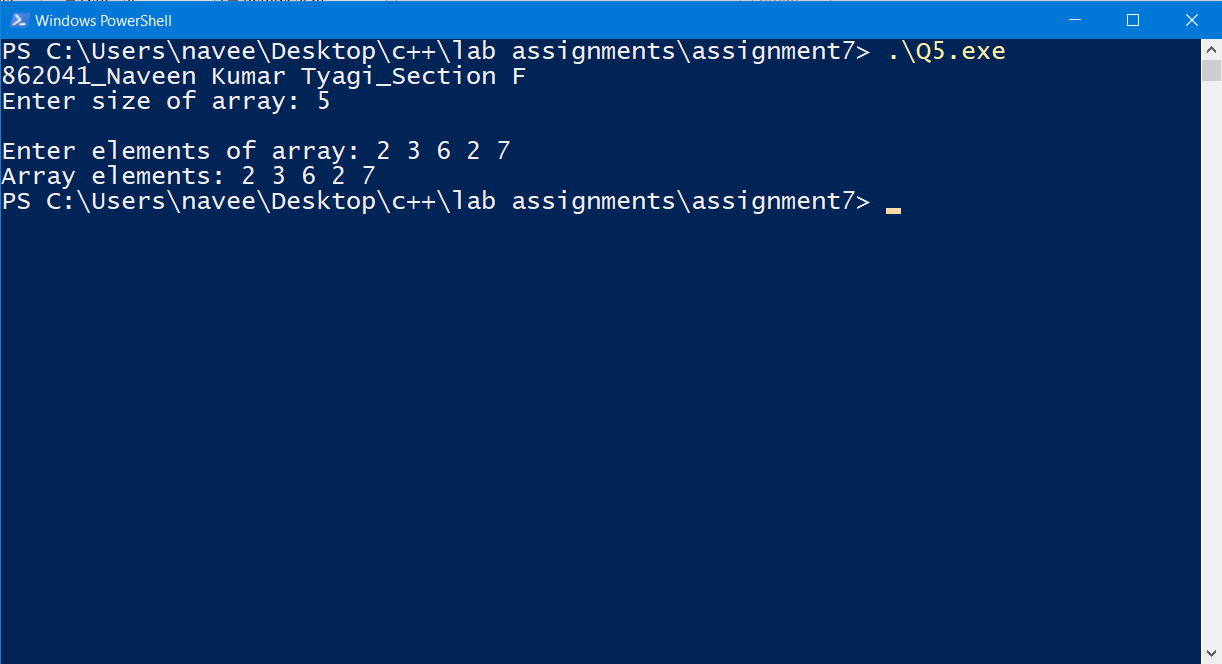
        cout<<\*(p+i)<<" "; //print out dereferenced value

    }

    return 0;

}

Output:



Q6. Let A and B be two arrays. Write a function to create a new array C that contains elements alternately from A and B beginning with the first element of A. Use pointer to access the elements from the array C. If you run out of elements in one of the lists (arrays), then append the remaining elements of the other list (array) to C.

Code:

//862041\_Naveen Kumar Tyagi\_Section F

#include<iostream>

using namespace std;

int main(){

    cout<<"862041\_Naveen Kumar Tyagi\_Section F\n";

    int a,b;

    cout<<"Enter size of array A: ";

    cin>>a;

    cout<<"Enter size of array B: ";

    cin>>b;

    int A[a],B[b],C[a+b];

    cout<<"Enter elements of Array A: ";

    //take input for array A

    for(int i=0;i<a;i++){

        cin>>A[i];

    }

    cout<<"Enter elements of Array B: ";

    //take input for array B

    for(int i=0;i<b;i++){

        cin>>B[i];

    }

    //elements of array A will stored at odd places in array C

    //elements of array B will stored at even places in array C

    //if size of array A is greater than array B

    //then after exhausion of array B

    //array A elements will be appended

    if(a>=b){

        //store A at odd places

        for(int i=0;i<b;i++){

            C[2\*i]=A[i];

        }

        //store B at even Places

        for(int i=0;i<b;i++){

            C[2\*i+1]=B[i];

        }

        //append A

        for(int i=0;i<a-b;i++){

            C[2\*b+i]=A[b+i];

        }

    }

    //if size of array B is greater than array A

    //then after exhausion of array A

    //array B elements will be appended

    else if(a<b){

        //store A at odd places

        for(int i=0;i<a;i++){

            C[2\*i]=A[i];

        }

        //store B at even Places

        for(int i=0;i<a;i++){

            C[2\*i+1]=B[i];

        }

        //append B

        for(int i=0;i<b-a;i++){

            C[2\*a+i]=B[a+i];

        }

    }

    //printing Array C using pointers

    int \*p=C;

    cout<<"\nArray C: ";

    for(int i=0;i<a+b;i++){

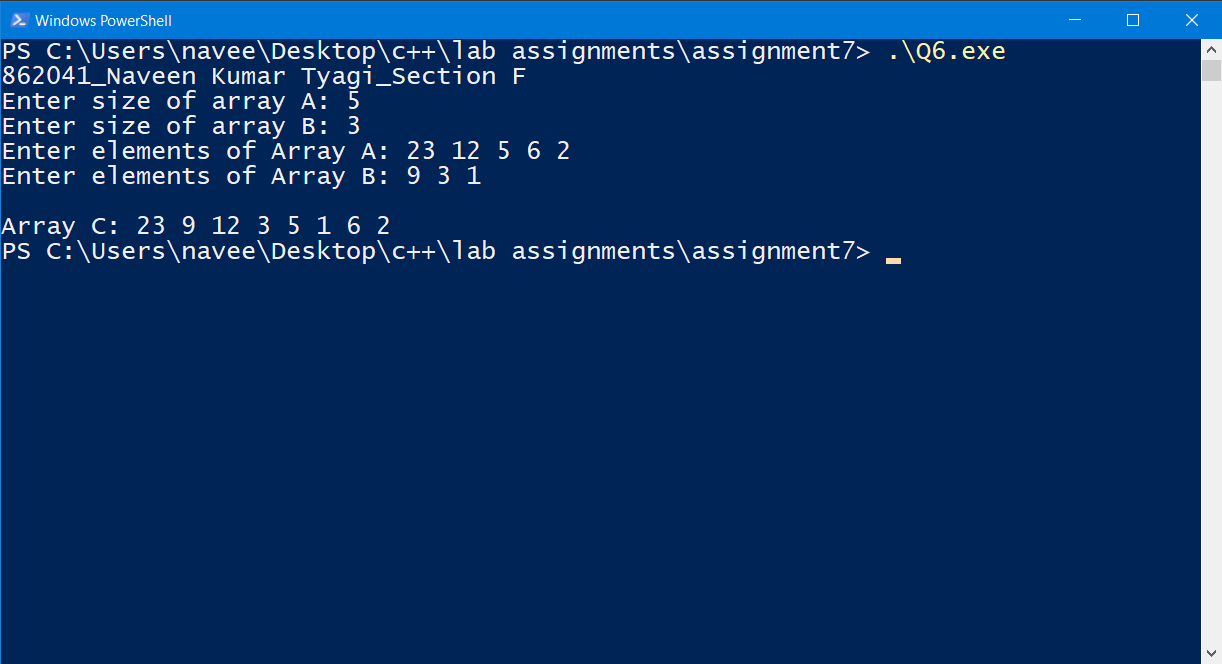
        cout<<\*(p+i)<<" ";  //print derefenced value

    }

    return 0;

}

Output:



Q7. Write a recursive program to multiply two given matrices, use the pointers to pass the matrices to the recursive function. Validate the size of the matrices in the main function.

Code:

//862041\_Naveen Kumar Tyagi\_Section F

#include<iostream>

using namespace std;

//recursive function to multiply matrix

void mat\_multplr(int \*A,int \*B,int \*C,int row1,int col1,int row2,int col2){

    //using static keyword, address are allocated to i,j,k for lifetime

    //of the program and is allocated only once

    //and values of variables in previous call carried to next call

    static int i=0,j=0,k=0;

    if(i<row1){

        if(j<col2){

            //this if statement block is to evaluate a row

            if(k<col1){

                //this if statement block is to evaluate a element

                //treating 2d array as a 1d array

                \*(C+i\*col2+j)+=(\*(A+i\*col1+k))\*(\*(B+k\*col2+j));

                k++;

                //(int\*) is used to assign address equal to that which is

                //written in right, since identifier(name) of array represents

                //address of first element of array,so,for eg.

                //(int\*)A assign address of first element of array

                mat\_multplr((int\*)A,(int\*)B,(int\*)C,row1,col1,row2,col2);

            }

            k=0;

            j++;//increase j by 1 so that next element of can be evaluated

            mat\_multplr((int\*)A,(int\*)B,(int\*)C,row1,col1,row2,col2);

        }

        j=0;

        i++;//increase i by 1 so that next row can be evaluated

        mat\_multplr((int\*)A,(int\*)B,(int\*)C,row1,col1,row2,col2);

    }

}

int main(){

    cout<<"862041\_Naveen Kumar Tyagi\_Section F\n";

    int row1,col1,row2,col2; //variables to store sizes of matrices

    cout<<"Enter size of Matrix A: ";

    cin>>row1>>col1;  //take input for size of matrix A

    cout<<"Enter size of Matrix B: ";

    cin>>row2>>col2;  //take input for size of matrix B

    if(col1==row2){   //condition whether multiplication is possible or not

        int A[row1][col1],B[row2][col2];  //declaration of the two matrices

        //taking input for Matrix A

        cout<<"Enter elements of Matrix A:\n";

        for(int i=0;i<row1;i++){

            for(int j=0;j<col1;j++){

                cin>>A[i][j];

            }

        }

        //taking input for Matrix B

        cout<<"Enter elements of Matrix B:\n";

        for(int i=0;i<row2;i++){

            for(int j=0;j<col2;j++){

                cin>>B[i][j];

            }

        }

        //initialising C matrix with all elements equal to 0

        int C[row1][col2]; //declaration of C matrix

        for(int i=0;i<row1;i++){

            for(int j=0;j<col2;j++){

                C[i][j]=0;

            }

        }

        //call of function to do matrix multiplication

        mat\_multplr((int\*)A,(int\*)B,(int\*)C,row1,col1,row2,col2);

        //printing out the product

        cout<<"Product matrix of given two matrices:\n";

        for(int i=0;i<row1;i++){

            for(int j=0;j<col2;j++){

                cout<<C[i][j]<<" ";

            }

            cout<<endl;

        }

    }

    //else block to tell multiplication is not possible

    //if required cond. for matrices multiplication not satisfied

    else{

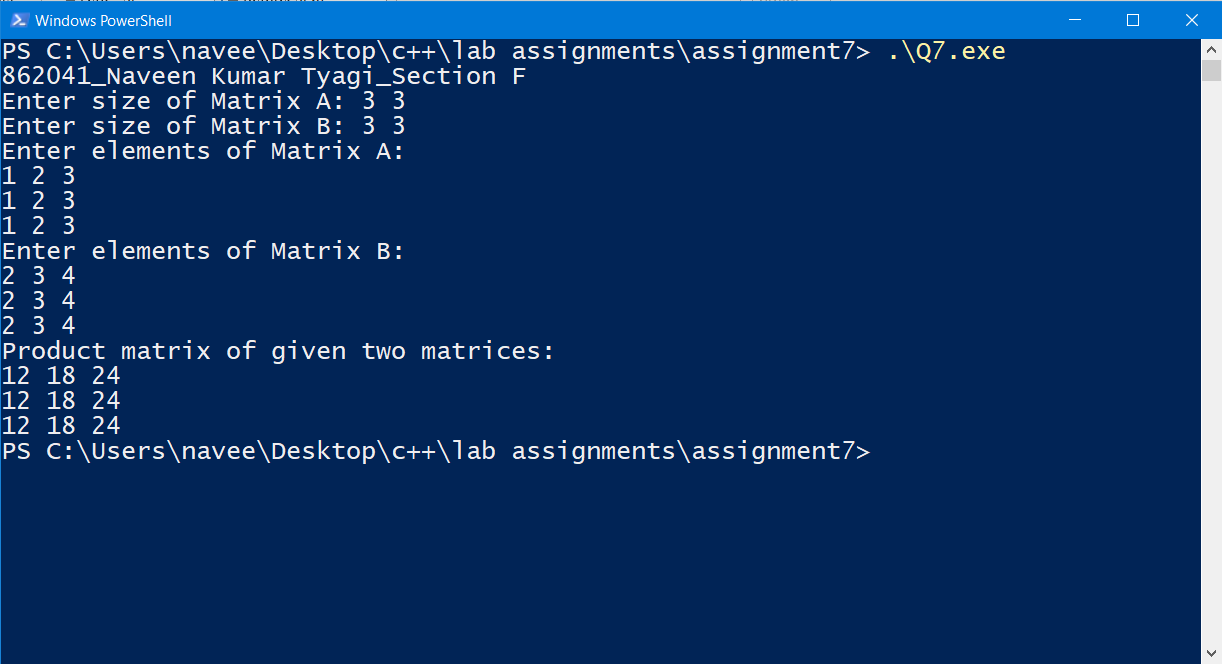
        cout<<"Matrix multiplication is not possible.";

    }

    return 0;

}

Output:



Q8. Write a program to read n integers into an array of int, to sort the elements in ascending order and to print the original and sorted elements. The program should not rearrange the elements of the array and should not create any other integer arrays. HINT - You may use two pointer arrays and rearrange the entries in these arrays.

Code:

//862041\_Naveen Kumar Tyagi\_Section F

#include<iostream>

using namespace std;

int main(){

    cout<<"862041\_Naveen Kumar Tyagi\_Section F\n";

    int n;

    cout<<"Enter size of array: ";

    cin>>n;

    int arr[n];

    cout<<"Enter elements of array: ";

    //take input and store in array

    for(int i=0;i<n;i++){

        cin>>arr[i];

    }

    //creating a pointer array that store addresses of input

    int \*ptr1[n];

    for(int i=0;i<n;i++){

        ptr1[i]=&arr[i];

    }

    //creating an another pointer array

    //elements of this array points to above pointer array elemnents

    int \*\*ptr2[n];

    for(int i=0;i<n;i++){

        ptr2[i]=&ptr1[i];

    }

    //compare input using 2nd pointer array

    //then arrange(sort) addresses of inputs in 1st pointer array

    //according to ascending order of input

    //sorting(insertion sort)

    for(int i=0;i<n-1;i++){

        int loc=i;

        int min=\*\*ptr2[i];

        for(int j=i;j<n;j++){

            if(min>\*\*ptr2[j]){

                min=\*\*ptr2[j];

                loc=j;

            }

        }

        //swapping the address

        int \*temp=\*ptr2[i];

        \*ptr2[i]=\*ptr2[loc];

        \*ptr2[loc]=temp;

    }

    //printing out original input

    cout<<"\nGiven elements: ";

    for(int i=0;i<n;i++){

        cout<<arr[i]<<" ";

    }

    //printing out sorted input

    cout<<"\nElements in ascending order: ";

    for(int i=0;i<n;i++){

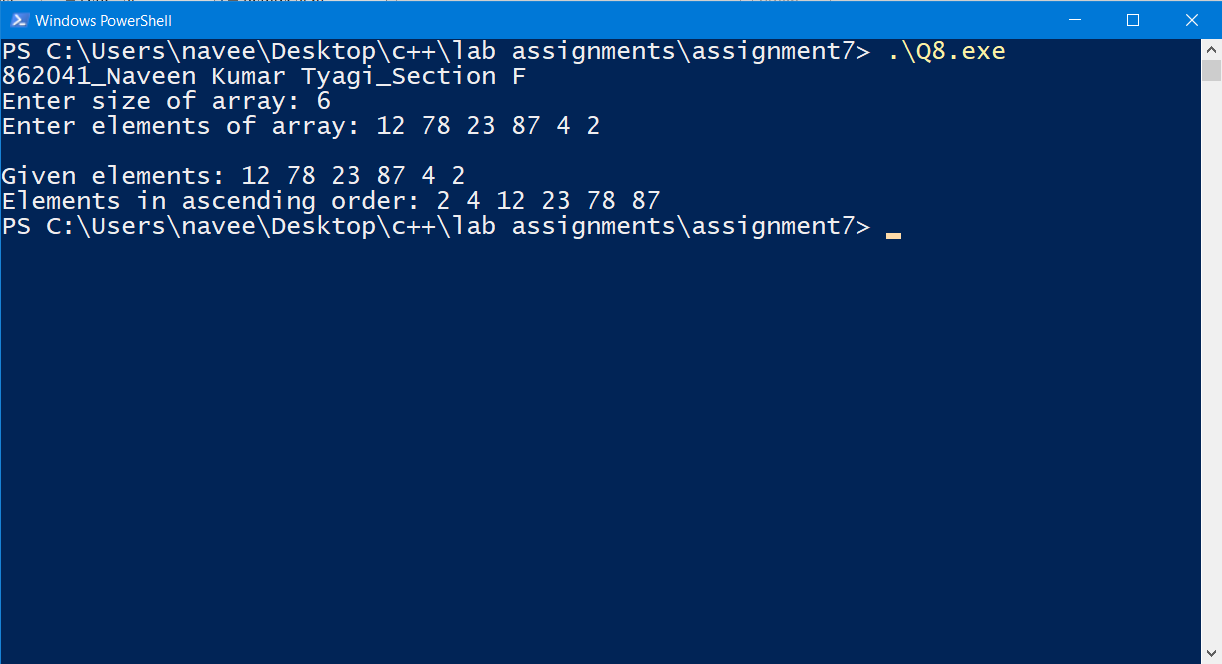
        cout<<\*ptr1[i]<<" ";

    }

    return 0;

}

Output:



Q9. Write a program to search for a given sub string inside a given string. If it is found in the string, then print the positions of all the occurrences.

Code:

//862041\_Naveen Kumar Tyagi\_Section F

#include<iostream>

#include<cstdio>

using namespace std;

//function to get the size of array

int size(char string[]){

    int i=0;

    //increase value of i by 1 until null character encountered

    while(string[i]!='\0'){

        i++;

    }

    //return i(size) finally

    return i;

}

//this function check the other characters of the substring in string

int checker(char string[],char sub\_string[],int size\_sub\_str,int index){

    for(int i=1;i<size\_sub\_str;i++){

        if(string[index+i]!=sub\_string[i]){

            return -1;  //return -1 if match not found

        }

    }

    return 1; //return 1 if all character matches

}

//this function find the match of first character of substring in string

//if match found then it pass the string, substring and location of match in string

//to checker function which then check other character

int finder(char string[],char sub\_string[],int size\_in\_str,int size\_sub\_str,int loc[10]){

    int ind=0;//to store number of matches(also used as index to store loc of matches)

    for(int i=0;i<size\_in\_str;i++){

        if(string[i]==sub\_string[0]){

            //res variable to store return value of checker function

            int res=checker(string,sub\_string,size\_sub\_str,i);

            //if complete match found then store location

            if(res==1){

                loc[ind++]=i+1;

            }

        }

    }

    //return ind, if it is 0 that it can be concluded that no match found

    return ind;

}

int main(){

    cout<<"862041\_Naveen Kumar Tyagi\_Section F\n";

    //declaration of arrays to store string and substring

    char input\_string[30],sub\_string[10];

    int loc[10]; //to store location of matches

    cout<<"Enter string: ";

    gets(input\_string); //store entered string in input\_string

    //calling size fucntion to get size of string and storing it

    int size\_in\_str=size(input\_string);

    cout<<"Enter substring: ";

    gets(sub\_string);  //store entered sub string in sub\_string

    //calling size fucntion to get size of substring and storing it

    int size\_sub\_str=size(sub\_string);

    //declaration of variable to store number of matches found

    //call finder function and store its return value

    int no\_find=finder(input\_string,sub\_string,size\_in\_str,size\_sub\_str,loc);

    if(no\_find==0){

        cout<<"There is no match.";

    }

    //if matches found then print the location of matches

    else{

        cout<<"\nLocations where substring found.\n";

        //print loc of matches

        for(int i=0;i<no\_find;i++){

            cout<<loc[i]<<" ";

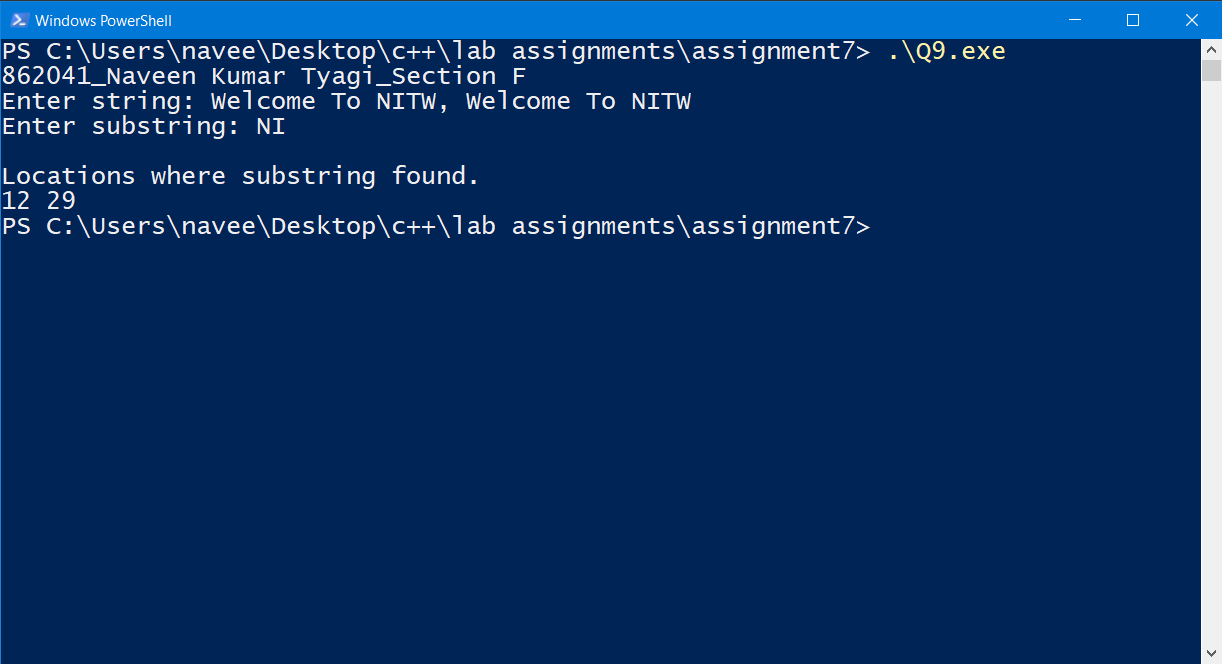
        }

    }

    return 0;

}

Output:



Q10. Write a program to convert string into lowercase and uppercase without using library function in C++

Code:

//862041\_Naveen Kumar Tyagi\_Section F

#include<iostream>

#include<cstdio>

using namespace std;

int main(){

    cout<<"862041\_Naveen Kumar Tyagi\_Section F\n";

    //declaration of required character arrays

    char input\_string[30],lower\_case[30], upper\_case[30];

    cout<<"Enter a string: ";

    gets(input\_string); //store entered string in input\_string(char array)

    //converting into lower case and upper case

    int i=0;

    while(input\_string[i]!='\0'){

        //if char is Capital then

        //store its small form in lower\_case array

        //store as it is in upper\_case array

        if(input\_string[i]>='A'&&input\_string[i]<='Z'){

            lower\_case[i]=input\_string[i]+32;

            upper\_case[i]=input\_string[i];

            i++;

        }

        //if char is small then

        //store as it is in lower\_case array

        //store its capital form in upper\_case array

        else if(input\_string[i]>='a'&&input\_string[i]<='z'){

            lower\_case[i]=input\_string[i];

            upper\_case[i]=input\_string[i]-32;

            i++;

        }

        //store spaces in both lower\_case and upper\_case array

        else if(input\_string[i]==' '){

            lower\_case[i]=input\_string[i];

            upper\_case[i]=input\_string[i];

            i++;

        }

    }

    //put null character at the end

    //of both lower\_case and upper\_case

    if(input\_string[i]=='\0'){

        lower\_case[i]='\0';

        upper\_case[i]='\0';

    }

    //prints entered string in lowercase

    cout<<"\nString in lowercase: "<<lower\_case;

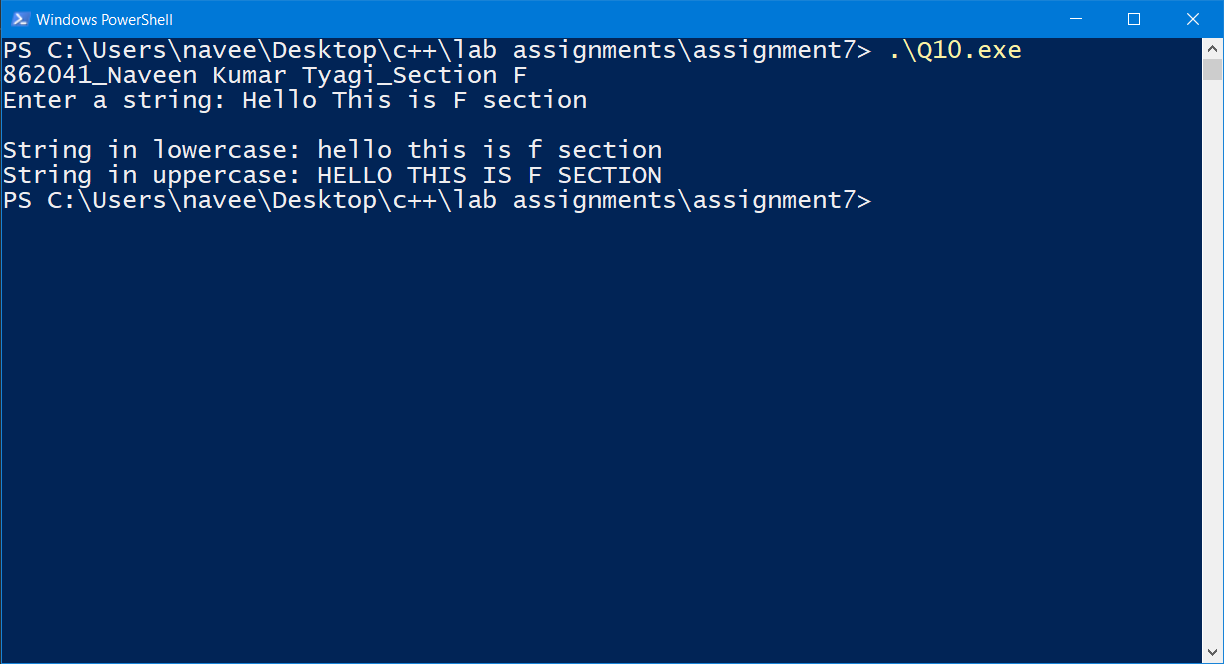
    //prints entered string in uppercase

    cout<<"\nString in uppercase: "<<upper\_case;

    return 0;

}

Output:



Q11. Write a program which calls a recursive function to determine the given string is palindrome or not and also write a recursive function to find the length the given string.

Code:

//862041\_Naveen Kumar Tyagi\_Section F

#include<iostream>

#include<string>

#include<cstdio>

using namespace std;

int len=0;//to store length of string

//recursive function to get length of string

int str\_len(char string[30],int len){

    //increase len one and call itself to do same

    //untill null character encountered

    if(string[len]!='\0'){

        len++;

        return str\_len(string,len);

    }

    //store value of local len in global len

    ::len=len;

    return len;

}

//recursive function to check whether string is palindrome is or not

//it check first char with last char

//if not match then it exit printing out 'not palindrome'

//or if it matches then it call itself

//and then check second with second last

//this will continue till middle char(if all matches)

//then it is palindrome

string palindrome\_checker(char string[30],int index,int mid,int len){

    if(index<mid){

        if(string[index]==string[(len-1)-index]){

            index++; //increase index to check next in recursive call

            return palindrome\_checker(string,index,mid,len);

        }

    }

    //if all matches then index will be equal to mid

    if(index==mid){

        return "\nString is palindrome.";

    }

    return "\nString is not palindrome.";

}

int main(){

    cout<<"862041\_Naveen Kumar Tyagi\_Section F\n";

    char input\_string[30];

    cout<<"Enter string: ";

    gets(input\_string); //store entered string in input\_string char array

    //printing the length of string by calling str\_len function

    cout<<"Length of String: "<<str\_len(input\_string,len);

    int mid=len/2;  //index of middle character

    //calling function palindome\_checker

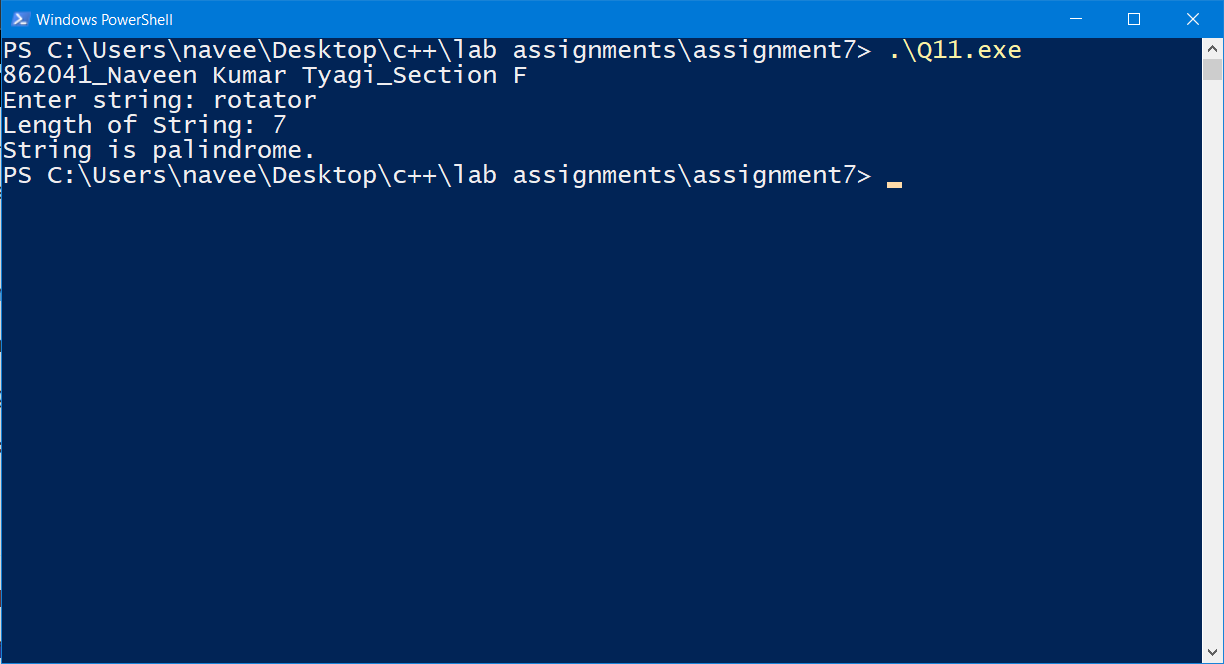
    //and printing whether entered string is palindrome or not

    cout<<palindrome\_checker(input\_string,0,mid,len);

    return 0;

}

Output:



Q12. Write a program to sort the given names in lexicographical order (Alphabetical-order).

Code:

//862041\_Naveen Kumar Tyagi\_Section F

#include<iostream>

#include<cstring>

using namespace std;

//function to take names form user

void names\_input(string names[],int n){

    for(int i=0;i<n;i++){

        cout<<"Enter name: ";

        getline(cin,names[i]);

    }

}

int main(){

    cout<<"862041\_Naveen Kumar Tyagi\_Section F\n";

    int n;  //variable to store number of names

    cout<<"Enter number of names: ";

    cin>>n;

    string names[n];  //string array to store names

    //cin.ignore() will clear characters from input buffer

    //so that getline() will function properly

    cin.ignore();

    //call of function to take desired number of names

    names\_input(names,n);

    //creating an pointer array

    //pointing to names

    string \*ptr\_names[n];

    for(int i=0;i<n;i++){

        ptr\_names[i]=&names[i];

    }

    //sorting (insertion sort) of address in pointer array

    //on the basis of names they are pointing

    //below code will sort in lexicographical order

    for(int i=0;i<3;i++){

        int loc=i;

        string min=\*ptr\_names[i];

        for(int j=i;j<4;j++){

            //strcmp is predifined

            //it compares string char by char

            //to do so, it needs character array

            //so c\_str() is used

            //it converts string to char array

            int res=strcmp(min.c\_str(),(\*ptr\_names[j]).c\_str());

            if(res>0){

                min=\*ptr\_names[j];

                loc=j;

            }

        }

        //swapping the addresses

        string \*temp=ptr\_names[i];

        ptr\_names[i]=ptr\_names[loc];

        ptr\_names[loc]=temp;

    }

    //print the given input in given order

    cout<<"Original Input(in given order): ";

    for(int i=0;i<n;i++){

        cout<<names[i]<<" ";

    }

    cout<<endl;

    //print the given input in lexicographical order

    cout<<"Names in lexicographical order: ";

    for(int i=0;i<n;i++){

        cout<<\*ptr\_names[i]<<" ";

    }

    return 0;

}

Output:

