DEPARTMENT OF COMPUTER ENGINEERING

Semester	T.E. Semester VI – Computer Engineering
Subject	Mobile Computing
Subject Professor In-charge	Prof. Sneha Annappanavar
Assisting Teachers	Prof. Sneha Annappanavar
Laboratory	M310A

Student Name	Deep Salunkhe
Roll Number	21102A0014
TE Division	A

Title:

Orthogonal Codes

Explanation:

In coding theory, orthogonal codes play a crucial role, particularly in communication systems. When two binary codes, A and B, are orthogonal, it means that the inner product of their corresponding sequences is zero.

Mathematically, for binary sequences A=(a1,a2,...,an) and B=(b1,b2,...,bn), the inner product is computed as:

$A \cdot B = \sum_{i=1}^{n} a_i \cdot b_i$

If $A \cdot B = 0$, then the codes A and B are orthogonal.

In our implementation, we utilize a bipolar representation of binary sequences, where '1' is represented as 1 and '0' is represented as -1. This bipolar representation is commonly used in communication systems.

Here's a breakdown of our code implementation:

1. Bipolar Function (Bipolar):
    - Converts a binary string to a bipolar sequence.
    - Each '1' is represented as 1, and each '0' is represented as -1.
2. Multiply Function (Multiply):
    - Performs element-wise multiplication of two bipolar sequences.

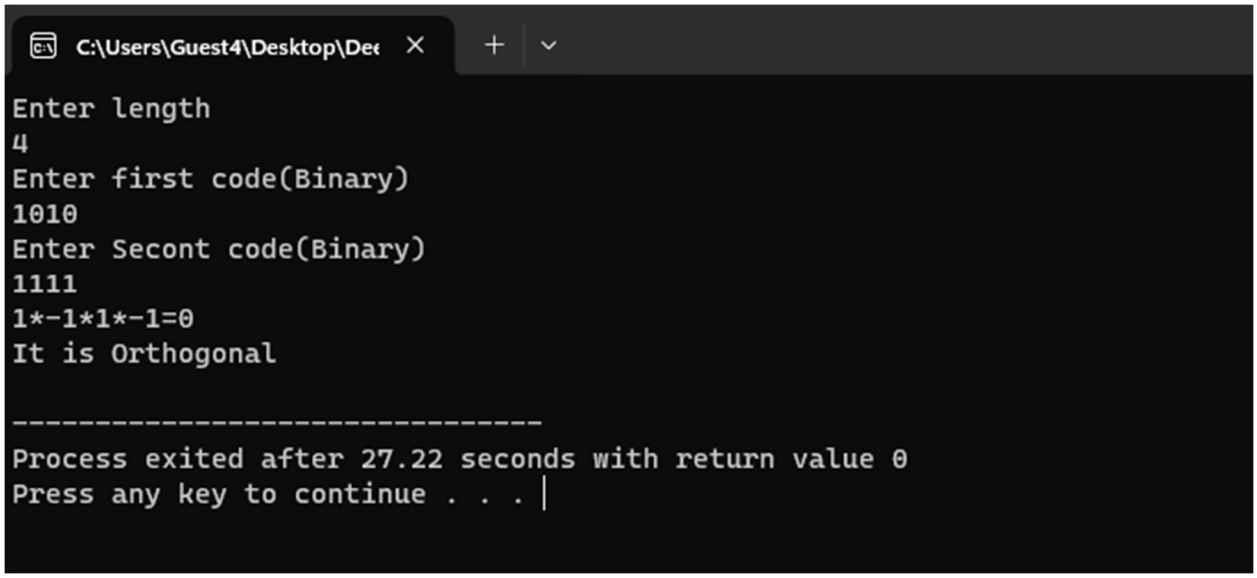
3. Main Function:
    - Takes two binary strings as input.
    - Converts them into bipolar sequences.
    - Performs element-wise multiplication of the sequences.
    - Checks if the sum of the multiplied sequence is zero to determine orthogonality.

Implementation:

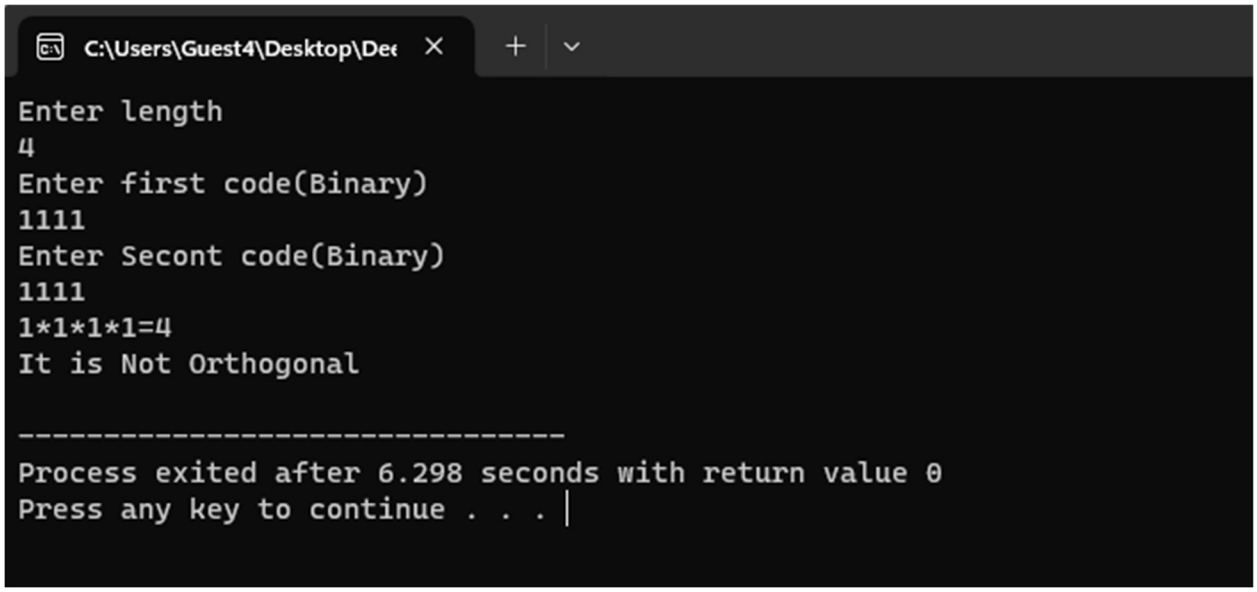
```
#include<iostream>
#include<vector>
using namespace std;
void Bipolar(vector<int>&C,string s){
    int size=s.length();
    for(int i=0;i<size;i++){
        if(s[i]=='1'){
            C.push_back(1);
        }else{
            C.push_back(-1);
        }
    }
}

void    Multiply(vector<int>A,vector<int>B,vector<int>&M){
    int size=A.size();
    for(int i=0;i<size;i++){
        M.push_back(A[i]*B[i]);
    }
}

void TakeCode(string &s,int size){

    int originalsize=size;

    cin>>s;
    if(s.length()!=size){
            cout<<"Invalid size "<<endl;
            return;
    }
    for(int i=0;i<size;i++){
        if(s[i]=='1'|| s[i]=='0'){
```

```
            continue;
     }else{
        cout<<"Not a Binary"<<endl;
        return;
     }

    }

}

int main(){
    int len=0;
    cout<<"Enter length"<<endl;
    cin>>len;
    string A,B;
    cout<<"Enter first code(Binary)"<<endl;
    TakeCode(A,len);
    cout<<"Enter Secont code(Binary)"<<endl;
    TakeCode(B,len);
    vector<int>AC,BC,M;
    Bipolar(AC,A);
    Bipolar(BC,B);
    Multiply(AC,BC,M);
    int temp=0;
    int size=A.size();
    for(int i=0;i<size;i++){
    temp=temp+M[i];
    }

    for(int i=0;i<len;i++){
        cout<<M[i];
        if(i!=len-1) cout<<"*";
    }
    cout<<"="<<temp<<endl;

    if(temp==0){
        cout<<"It is Orthogonal"<<endl;

    }else
    {
            cout<<"It is Not Orthogonal"<<endl;
    }
    return 0;
}
```

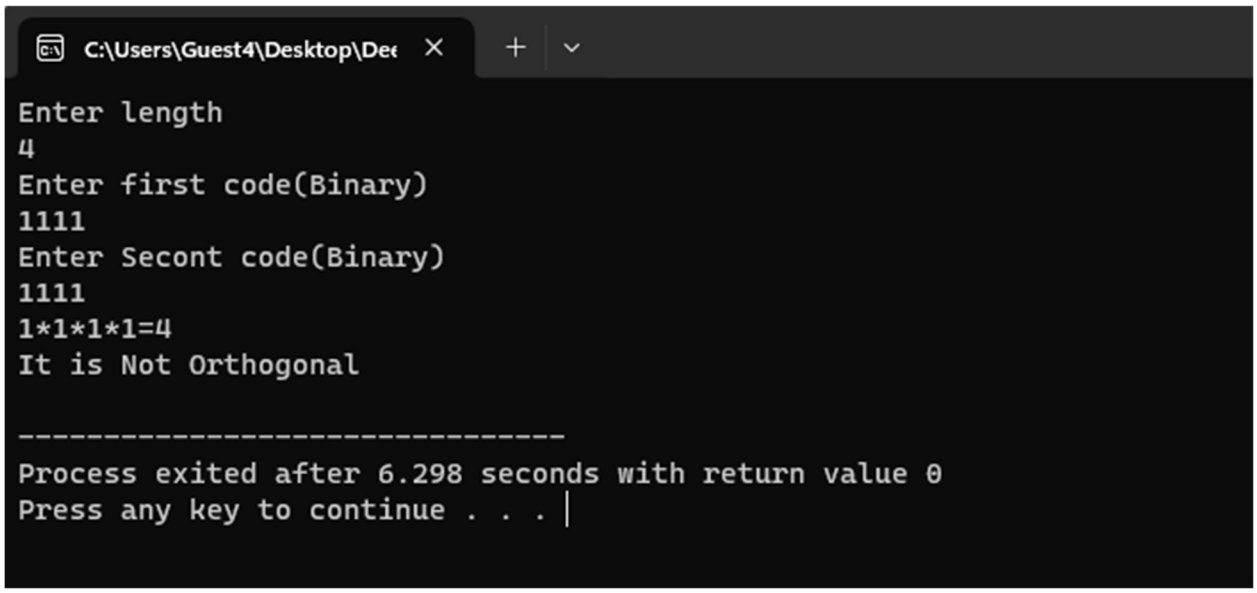
End Result:


C:\Users\Guest4\Desktop\Dee
Enter length
4
Enter first code(Binary)
1010
Enter Secont code(Binary)
1111
1*-1*1*-1=0
It is Orthogonal

Process exited after 27.22 seconds with return value 0
Press any key to continue . . .


C:\Users\Guest4\Desktop\Dee
Enter length
4
Enter first code(Binary)
1111
Enter Secont code(Binary)
1111
1*1*1*1=4
It is Not Orthogonal

Process exited after 6.298 seconds with return value 0
Press any key to continue . . .



Conclusion:

The code essentially checks whether the two input binary codes are orthogonal by converting them to bipolar sequences and performing element-wise multiplication. If the sum of the multiplied sequence is zero, the codes are orthogonal.