INTRODUCTION TO PLOTTING COMMANDS

LAB # 04

CSE402L Digital Signal Processing Lab

Submitted by: **Shah Raza**

Registration No: **18PWCSE1658**

Class Section: **B**

"On my honor, as a student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Student Signature: ______________

Submitted to: **Engr. Faiz Ullah**

Wednesday, December 30th, 2020

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

Lab Objectives:

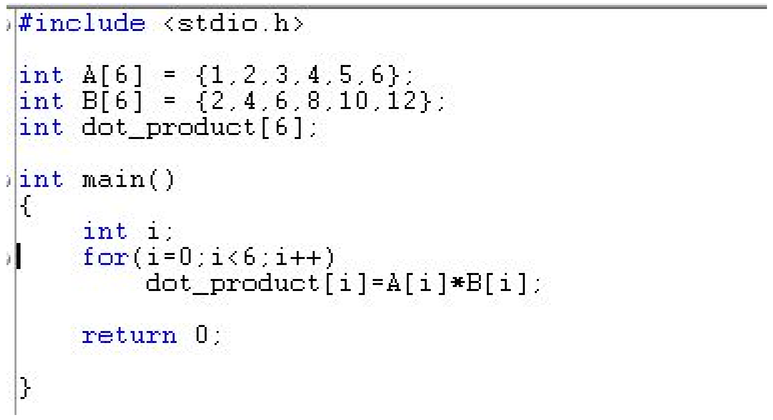
Objectives of this lab are as follows:

- Learn plotting in Visual DSP++

Practical:

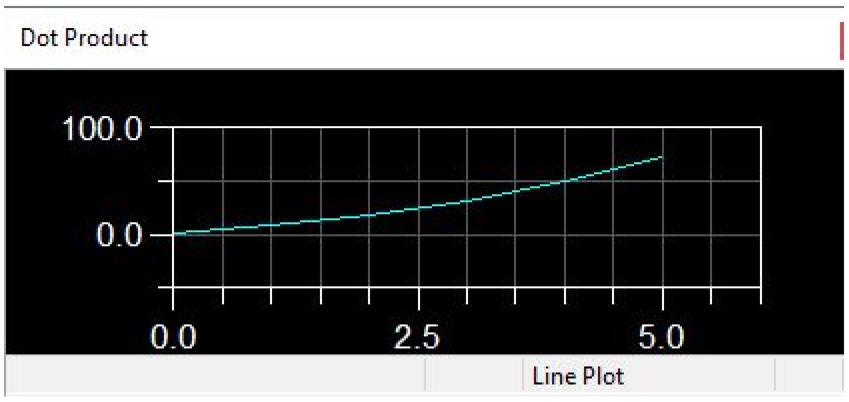
Generate two arrays of type int in VisualDSP++ 4.5. Consider these arrays as vectors and perform the dot product between the two arrays.

Code:



```
#include <stdio.h>

int A[6] = {1,2,3,4,5,6};
int B[6] = {2,4,6,8,10,12};
int dot_product[6];

int main()
{
	int i;
	for(i=0;i<6;i++)
		dot_product[i]=A[i]*B[i];

	return 0;

}
```

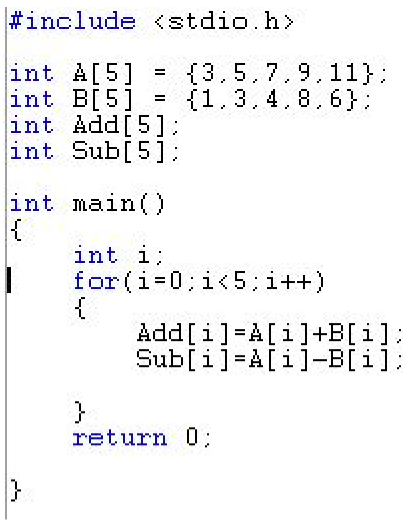
Output:



Task # 1:

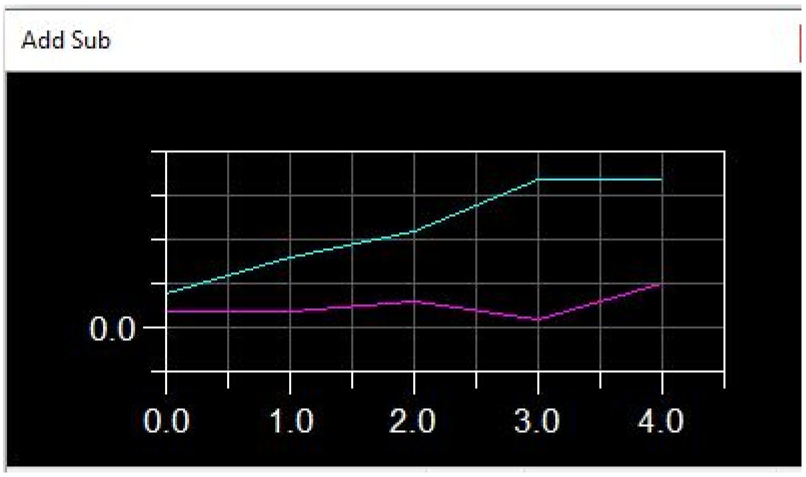
Add and Subtract two vectors, element by element. Plot the output vector.

Code:



```c
#include <stdio.h>

int A[5] = {3,5,7,9,11};
int B[5] = {1,3,4,8,6};
int Add[5];
int Sub[5];

int main()
{
    int i;
    for(i=0;i<5;i++)
    {
        Add[i]=A[i]+B[i];
        Sub[i]=A[i]-B[i];

    }
    return 0;

}
```

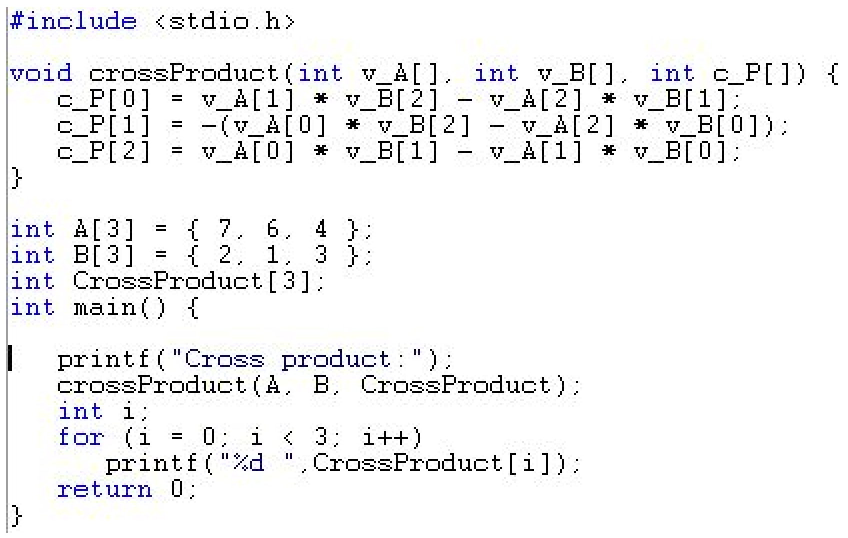
Output:



Task # 2:

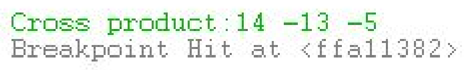
Perform the cross product of two integer arrays.

Code:

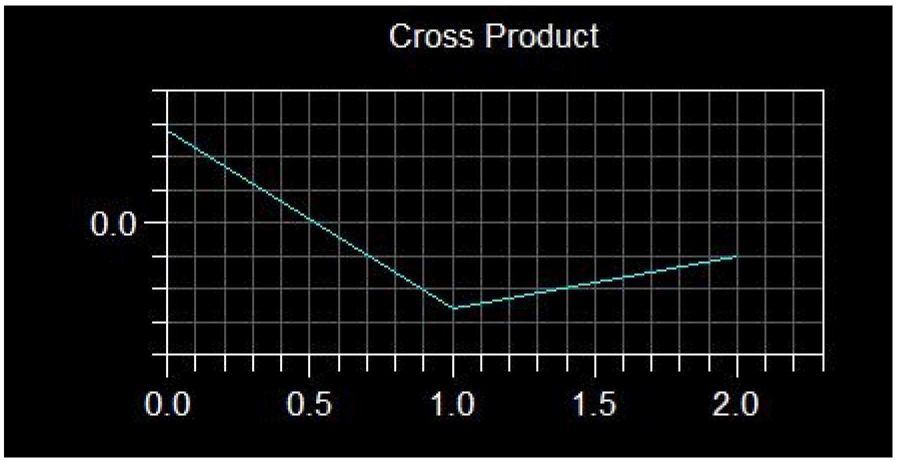


```c
#include <stdio.h>

void crossProduct(int v_A[], int v_B[], int c_P[]) {
   c_P[0] = v_A[1] * v_B[2] - v_A[2] * v_B[1];
   c_P[1] = -(v_A[0] * v_B[2] - v_A[2] * v_B[0]);
   c_P[2] = v_A[0] * v_B[1] - v_A[1] * v_B[0];
}

int A[3] = { 7, 6, 4 };
int B[3] = { 2, 1, 3 };
int CrossProduct[3];
int main() {

   printf("Cross product:");
   crossProduct(A, B, CrossProduct);
   int i;
   for (i = 0; i < 3; i++)
      printf("%d ",CrossProduct[i]);
   return 0;
}
```

Output:



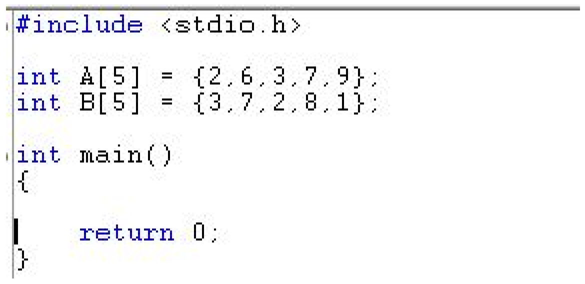
```
Cross product:14 -13 -5
Breakpoint Hit at <ffa11382>
```



Task # 3:

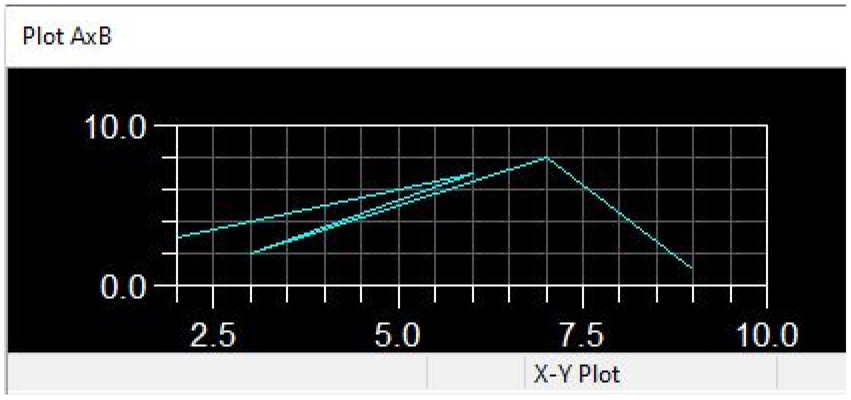
Plot one vector with respect to the other vector.

Code:



```
#include <stdio.h>

int A[5] = {2,6,3,7,9};
int B[5] = {3,7,2,8,1};

int main()
{

    return 0;
}
```

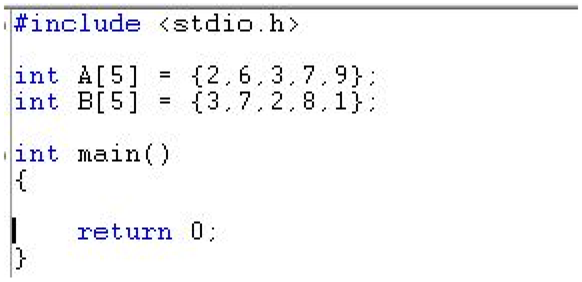
Output:



Task # 4:

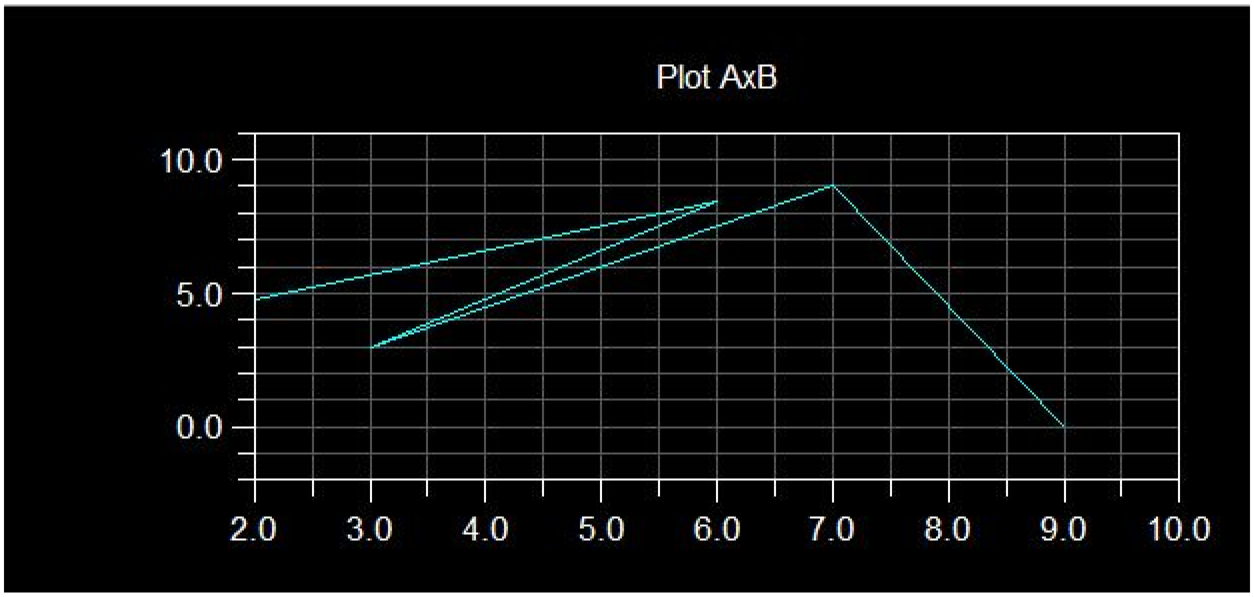
In the above procedure, convert the plot display to dB scale.

Code:



```
#include <stdio.h>

int A[5] = {2,6,3,7,9};
int B[5] = {3,7,2,8,1};

int main()
{

    return 0;
}
```

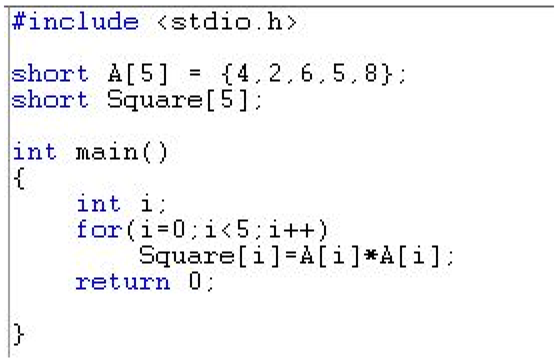
Output:



Task # 5:

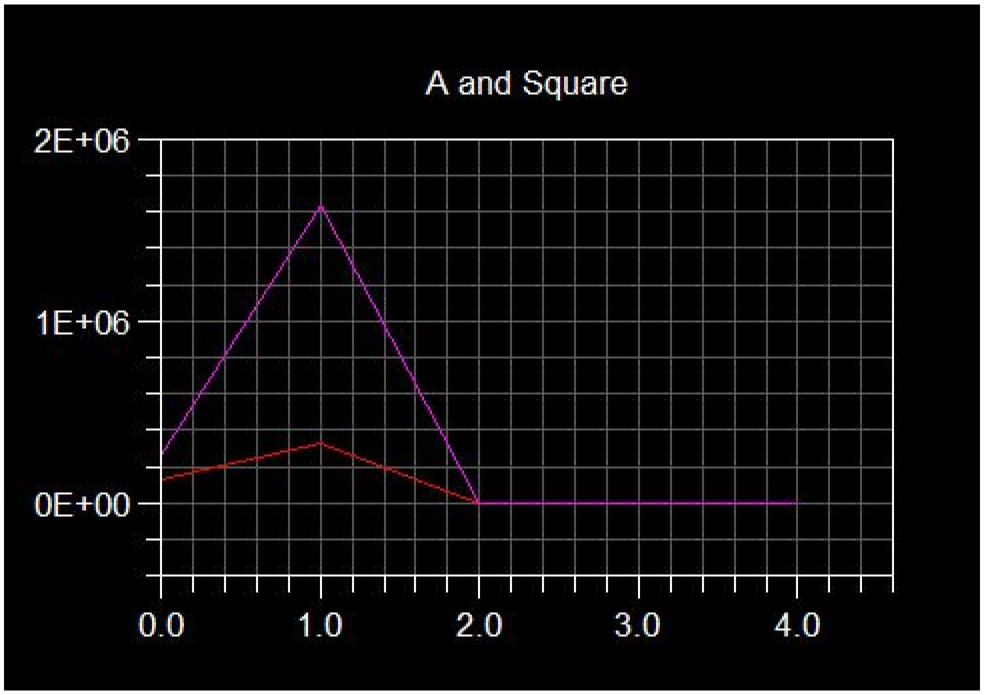
Define a short type vector a. Square each of its elements. Display the vector and its Square on the same plot with different color schemes.

Code:



```c
#include <stdio.h>

short A[5] = {4,2,6,5,8};
short Square[5];

int main()
{
    int i;
    for(i=0;i<5;i++)
        Square[i]=A[i]*A[i];
    return 0;

}
```

Output:



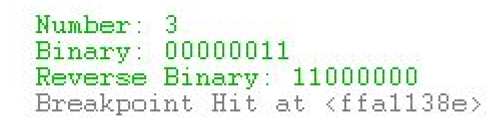
Task # 6:

Write a program in VisualDSP to reverse the order of bits in an 8-bit variable.

Code:

```
#include <stdio.h>
void Binary(char ch)
{

        int i;
        for(i=7;i>=0;i--)
        {
                if(ch&(1<<i))
                        printf("1");
                else
                        printf("0");

        }
}
int main()
{
        int a=3;
        printf("Number: %d\n",a);
        printf("Binary: ");
        Binary((char)a);
        printf("\nReverse Binary: ");
        int i;
        char temp,rev=0;
        for(i=0;i<8;i++)
        {
                temp=((char)a&(1<<i))>>i;
                temp=temp<<(7-i);
                rev=(rev|temp);

        }
        Binary(rev);
        return 0;
}
```

Output:



```
Number: 3
Binary: 00000011
Reverse Binary: 11000000
Breakpoint Hit at <ffa1138e>
```