Write a procedure, called dot\_product which calculates in the variable ps, the dot(scalar) product of v1 and v2 (v1 and v2 are vectors of IR)

Write an algorithm which determines, for n pairs of given vectors, whether two vectors of given IR are orthogonal, by calling the procedure defined in the previous question. The dot product of two orthogonal vectors is zero.using namespace std;

// Function that return

// dot product of two vector array.

int dotProduct(int vect\_A[], int vect\_B[])

{

int product = 0;

// Loop for calculate dot product

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

product = product + vect\_A[i] \* vect\_B[i];

return product;

}

// Function to find

// cross product of two vector array.

void crossProduct(int vect\_A[], int vect\_B[], int cross\_P[])

{

cross\_P[0] = vect\_A[1] \* vect\_B[2] - vect\_A[2] \* vect\_B[1];

cross\_P[1] = vect\_A[2] \* vect\_B[0] - vect\_A[0] \* vect\_B[2];

cross\_P[2] = vect\_A[0] \* vect\_B[1] - vect\_A[1] \* vect\_B[0];

}

// Driver function

int main()

{

int vect\_A[] = { 3, -5, 4 };

int vect\_B[] = { 2, 6, 5 };

int cross\_P[n];

// dotProduct function call

cout << "Dot product:";

cout << dotProduct(vect\_A, vect\_B) << endl;

// crossProduct function call

cout << "Cross product:";

crossProduct(vect\_A, vect\_B, cross\_P);

// Loop that print

// cross product of two vector array.

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

cout << cross\_P[i] << " ";

return 0;

}