

Diana Kerim ACDA4003

by Fatma Abd al kerim

FILE	10706_FATMA_ABD_AL_KERIM_DIANA_KERIM_ACDA4003_250776_52872 7641.DOCX (4.22M)		
TIME SUBMITTED	17-JUN-2020 12:37AM (UTC+0100)	WORD COUNT	3027
SUBMISSION ID	130795667	CHARACTER COUNT	14481

MATHEMATICS

Fatma Diana Abd Al Kerim – 1803845

ACDA4003

Contents

1. Introduction	2
1.1. Declaration	2
1.2. Abstract	2
2. Operations with Worked Examples	2
2.1. Find and Return the Length of a Vector	2
2.1.1. Find and Return the Length of a 2D Vector	2
2.1.2. Find and Return the Length of a 3D Vector	2
2.2. Normalize a Given Vector	3
2.3. Distance Between Two Points	4
2.4. Dot Product of Two Vectors	5
2.5. Cross Product of Two Vectors	6
2.7. Mean, Mode and Standard Deviation of a List of Integers	7
2.7.1. Mean of a List of Integers	Error! Bookmark not defined.
2.7.2. Mode of a List of Integers	Error! Bookmark not defined.
2.7.3. Standard Deviation of a List of Integers	Error! Bookmark not defined.
3. Completed Functions/Methods	8
3.1. Find and Return the Length of a Vector	8
3.2. Normalize a Given Vector	10
3.3. Distance Between Two Points	12
3.4. Dot Product of Two Vectors	13
3.5. Cross Product of Two Vectors	14
3.7. Mean of a List of Integers	15
6. References	16
7. Appendices	17
7.1. Appendix 1: Testing Schedule for Finding and returning the length of a Vector	17
7.2. Appendix 2: Testing Schedule for Normalize a given vector	20
7.3. Appendix 3: Testing Schedule for Finding the distance between two points	21
7.4. Appendix 4: Testing Schedule for calculating the dot product of two vectors	21
7.5. Appendix 5: Testing Schedule for calculating the cross product of two vectors	21
7.6. Appendix 6: Testing Schedule for calculating the Mean	22
7.7. Appendix 7: Code for Complete Program	22

1. Introduction

1.1. Declaration

The work in this assignment was completed by Fatma Diana Abd Al Kerim.

1.2. Abstract

In this assignment various operations and methods will be described, explained with worked examples, and coded into functions to work within a new software application. Following at the end, is a test schedule which verifies that the operations and methods are working as expected and the results are satisfactory.

2. Operations with Worked Examples

2.1. Find and Return the Length of a Vector

To find the length of a vector you use Pythagoras' theorem.

2.1.1. Find and Return the Length of a 2D Vector

Formula: $|a| = \sqrt{x^2 + y^2}$

Worked Example:

* Worked example for 2D Vector

1. $\sqrt{6^2 + 4^2}$	(calculate each number to the power of two)
2. $\sqrt{36 + 16}$	(Add the numbers together)
3. $\sqrt{52}$	(Square root the result)
4. ≈ 7.21	(this is the length of the 2D vector)

2.1.2. Find and Return the Length of a 3D Vector

Formula: $|a| = \sqrt{x^2 + y^2 + z^2}$

Worked Example:

* Solution (worked example)

1. $\sqrt{6^2 + 4^2 + 2^2}$	$\sqrt{x^2 + y^2 + z^2}$	(calculate each number to the power of two)
2. $\sqrt{36 + 16 + 4}$		(Add together)
3. $\sqrt{56}$		(square root)
4. ≈ 7.48		(Result)

2.2. Normalize a Given Vector

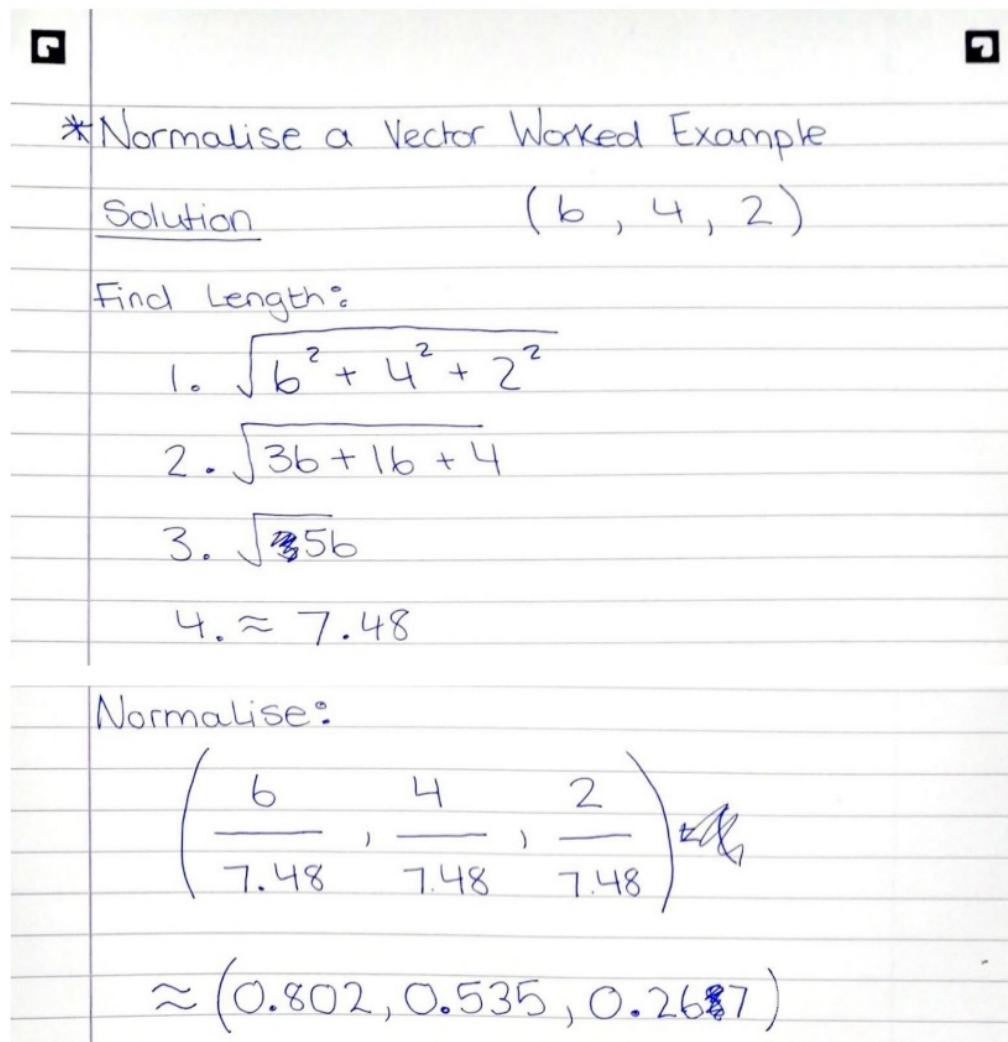
To normalise a vector, you must first find its length and then divide each element of the vector by its length and the result is a normalised vector.

Formula to normalise a 3-dimensional vector:

First find the length: $|a| = \sqrt{x^2 + y^2 + z^2}$

Divide each element by its length: $\langle x/a, y/a, z/a \rangle$

Worked Example:



The image shows a handwritten worked example on lined paper. It starts with the title '*Normalise a Vector Worked Example'. Below it, the vector (6, 4, 2) is written. The word 'Solution' is underlined. The first step is 'Find Length:'. This is followed by four numbered steps: 1. $\sqrt{6^2 + 4^2 + 2^2}$, 2. $\sqrt{36 + 16 + 4}$, 3. $\sqrt{56}$, and 4. ≈ 7.48 . The next step is 'Normalise:'. This is followed by the expression $\left(\frac{6}{7.48}, \frac{4}{7.48}, \frac{2}{7.48} \right)$ with a small arrow pointing to the right. Finally, the result is given as $\approx (0.802, 0.535, 0.2687)$.

*Normalise a Vector Worked Example

Solution (6, 4, 2)

Find Length:

1. $\sqrt{6^2 + 4^2 + 2^2}$
2. $\sqrt{36 + 16 + 4}$
3. $\sqrt{56}$
4. ≈ 7.48

Normalise:

$$\left(\frac{6}{7.48}, \frac{4}{7.48}, \frac{2}{7.48} \right)$$
$$\approx (0.802, 0.535, 0.2687)$$

2.3. Distance Between Two Points

The distance between two points is calculated by taking away the value of the second vector from the corresponding value in the first vector and squaring the result. All the results are then added together to final square root the total, the resulting value is the distance between the two points.

Formula to calculate the distance between two points in 3-dimensional space:

$$\text{Distance} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

Worked Example:

Distance between two points
Worked Example.

$(2, 4, 6)$ and $(1, 3, 5)$

Formula = $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$

$= \sqrt{(2 - 1)^2 + (4 - 3)^2 + (6 - 5)^2}$

$= \sqrt{1^2 + 1^2 + 1^2}$

$= \sqrt{3}$

≈ 1.732

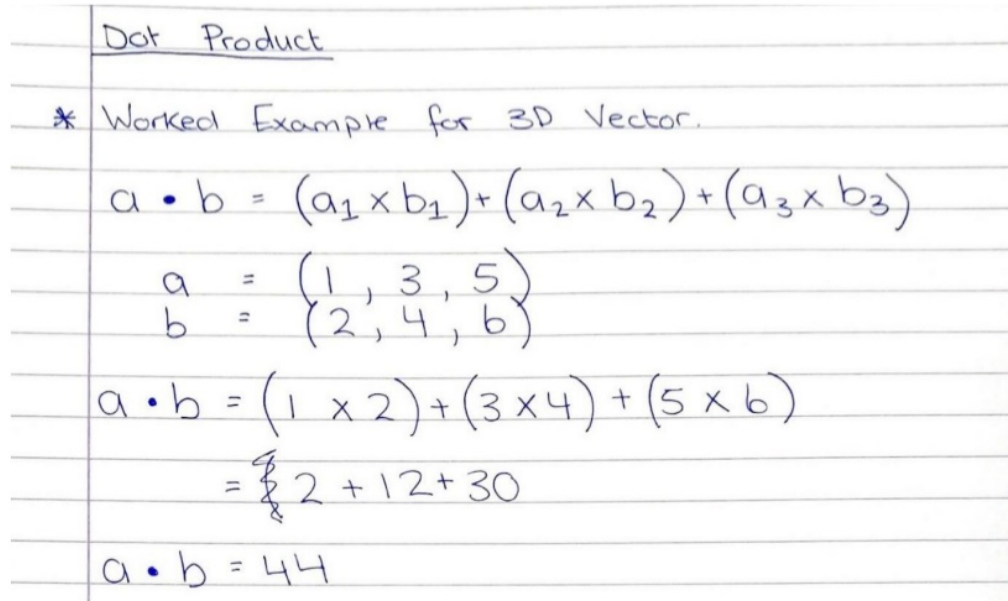
2.4. Dot Product of Two Vectors

To calculate the dot product of two vectors you take each corresponding coordinate of each vector and multiply them by each other, adding all the results together will give the dot product of those two vectors.

Formula to calculate the Dot Product of Two 3-Dimensional Vectors:

$$\text{Dot Product} = A \cdot B = (A_1 \times B_1) + (A_2 \times B_2) + (A_3 \times B_3)$$

Worked Example:



Handwritten worked example for 3D vector dot product:

Dot Product

* Worked Example for 3D Vector.

$$a \cdot b = (a_1 \times b_1) + (a_2 \times b_2) + (a_3 \times b_3)$$
$$a = (1, 3, 5)$$
$$b = (2, 4, 6)$$
$$a \cdot b = (1 \times 2) + (3 \times 4) + (5 \times 6)$$
$$= 2 + 12 + 30$$
$$a \cdot b = 44$$

2.5. Cross Product of Two Vectors

Calculating the cross product of two 3-dimensional vectors yields a third vector which is at a 90-degree angle of the two original vectors.

Formula:

The image shows a handwritten formula for the cross product of two vectors, a and b , resulting in vector C . The title "Cross Product" is underlined. The formulas for the components are:

$$C_1 = (a_2 \times b_3) - (a_3 \times b_2)$$
$$C_2 = (a_3 \times b_1) - (a_1 \times b_3)$$
$$C_3 = (a_1 \times b_2) - (a_2 \times b_1)$$

Worked Example:

The image shows a handwritten worked example for the cross product. The title "Worked Example" is underlined. The vectors a and b are given as $a = (2, 3, 4)$ and $b = (5, 6, 7)$. The components of the resulting vector C are calculated as follows:

$$C_1 = (3 \times 7) - (4 \times 6) = (21 - 24) = (-3)$$
$$C_2 = (4 \times 5) - (2 \times 7) = (20 - 14) = 6$$
$$C_3 = (2 \times 6) - (3 \times 5) = (12 - 15) = (-3)$$

The final result is $C = (-3, 6, -3)$.

2.7. Mean of a List of Integers

Formula:

Worked Example:

[Upload Note](#)

3. Completed Functions/Methods

3.1. Find and Return the Length of a Vector

Code Solution Prior To Coding (Calculate the length of a vector):

```
* LENTH
for (int i = 0; i < dimensions; i++)
{
    Vector[i] = Vector[i] * Vector[i];
    Answer = Answer + Vector[i];
}
Answer = sqrt(Answer);
cout << Answer;
```

Program Solution (Calculate the length of a vector):

```
22 void Length() {
23     for (int i = 0; i < dimensions; i++)
24     {
25         Vector[i] = Vector[i] * Vector[i];
26         Answer = Answer + Vector[i];
27     }
28     Answer = sqrt(Answer);
29     cout << "Result: " << Answer << endl;
30     system("pause");
31 }
32
33
34
35
36
```

On line 24 starts a 'for' loop which loops for a set number of times depending on the number of dimensions that the user has entered to compute.

Within the for loop is a calculation which will multiply each element of the array by itself, then each multiplication is added to a running total.

Once the for loop has terminated the answer of the total multiplications of

each element is then square rooted (line 30) and output to the user (line 32).

Testing (Calculate the length of a vector):

To test the program, six calculations were performed to check for the correct output. Half were to test the input of a 2-dimensional vector and the other half were to test for the correct output for a 3-dimensional vector.

Details of expected results, actual results and screenshots for all tests conducted can be located in Appendix 1.

Test Number	Description	Pass/Fail
1.1	To return the length of a vector, in this test we are checking for the correct return of the 2-dimensional vector of... $\vec{a} = (77 \ 79)$	PASS
1.2	To return the length of a vector, in this test we are checking for the correct return of the 2-dimensional vector of...	PASS

	$\vec{a} = \begin{pmatrix} 13 & 58 \end{pmatrix}$	
1.3	To return the length of a vector, in this test we are checking for the correct return of the 2-dimensional vector of... $\vec{a} = \begin{pmatrix} 89 & 65 \end{pmatrix}$	PASS
2.1	To return the length of a vector, in this test we are checking for the correct return of the 3-dimensional vector of... $\vec{a} = \begin{pmatrix} 26 & 24 & 98 \end{pmatrix}$	PASS
2.2	To return the length of a vector, in this test we are checking for the correct return of the 3-dimensional vector of... $\vec{a} = \begin{pmatrix} 87 & 46 & 76 \end{pmatrix}$	PASS
2.3	To return the length of a vector, in this test we are checking for the correct return of the 3-dimensional vector of... $\vec{a} = \begin{pmatrix} 29 & 67 & 34 \end{pmatrix}$	PASS

3.2. Normalize a Given Vector

Code Solution Prior To Coding (Normalize a Vector):

```

* NORM
for (int i = 0; i < dimensions; i++) {
    VectorA[i] = Vector[i] * Vector[i];
    Answer = Answer + VectorA[i];
}
Answer = Sqrt(Answer)

for (int i = 0; i < dimensions; i++) {
    Vector[i] = Vector[i] / Answer;
} cout << "(";
for (int i = 0; i < dimensions; i++) {
    cout << Vector[i] << ", ";
} cout << ")";

```

Program Solution (Normalize a Vector):

```

38 void Norm() {
39
40     for (int i = 0; i < dimensions; i++)
41     {
42         VectorA[i] = Vector[i] * Vector[i];
43         Answer = Answer + VectorA[i];
44     }
45
46     Answer = sqrt(Answer);
47
48     for (int i = 0; i < dimensions; i++)
49     {
50         Vector[i] = Vector[i] / Answer;
51     }
52
53     cout << "Result : (";
54
55     for (int i = 0; i < dimensions; i++)
56     {
57         cout << Vector[i] << ", ";
58     }
59
60     cout << ")";
61
62     system("pause");
63
64 }

```

Testing (Normalise a Vector):

Details of expected results, actual results and screenshots for all tests conducted can be located in appendix 2.

Test Number	Description	Pass/Fail
1.1	To return a normalised vector, in this test we are checking for the correct normalisation of the 3-dimensional vector of... $\left \vec{a} \right = \left (26 \ 24 \ 98) \right $	PASS
1.2	To return the length of a vector, in this test we are checking for the correct normalisation of the 3-dimensional vector of... $\left \vec{a} \right = \left (87 \ 46 \ 76) \right $	PASS
1.3	To return the length of a vector, in this test we are checking for the correct normalisation of the 3-dimensional vector of... $\left \vec{a} \right = \left (29 \ 67 \ 34) \right $	PASS

3.3. Distance Between Two Points

Code Solution Prior To Coding:

~~* Dot Product DISTANCE~~

```

for (int i = 0; i < Dimensions; i++)
{
    Vector[i] = VectorA[i] - VectorB[i];
    Vector[i] = sqrt(Vector[i]);
    Vector[i] = Vector[i] * Vector[i];
    Answer = Answer + Vector[i];
    Answer = sqrt(answer);
}

Answer = sqrt(answer);

cout << Answer;

```

Program Solution:

```

66 void Distance() {
67
68     for (int i = 0; i < dimensions; i++)
69     {
70
71         Vector[i] = VectorA[i] - VectorB[i];
72         Vector[i] = Vector[i] * Vector[i];
73         Answer = Answer + Vector[i];
74
75     }
76
77     Answer = sqrt(Answer);
78
79     cout << "Result: " << Answer;
80
81     system("pause");
82
83 }

```

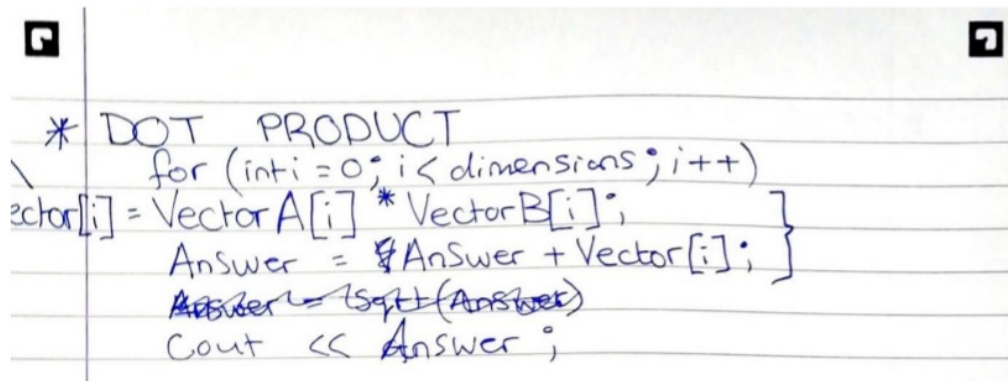
Testing:

Details of expected results, actual results and screenshots for all tests conducted can be located in appendix 3.

Test Number & Name	Description	Pass/Fail

3.4. Dot Product of Two Vectors

Code Solution Prior To Coding:



Program Solution:

```

85 void DotProduct() {
86
87     for (int i = 0; i < dimensions; i++)
88     {
89         Vector[i] = VectorA[i] * VectorB[i];
90         Answer = Answer + Vector[i];
91     }
92
93     cout << Answer;
94
95 }

```

Testing:

Details of expected results, actual results and screenshots for all tests conducted can be located in appendix 4.

Test Number & Name	Description	Pass/Fail

3.5. Cross Product of Two Vectors

Code Solution Prior To Coding:

Cross Product - Ruff code

$$\text{Vector}[0] = (\text{VectorA}[1] * \text{VectorB}[2]) - (\text{VectorA}[2] * \text{VectorB}[1])$$

$$\text{Vector}[1] = (\text{VectorA}[2] * \text{VectorB}[0]) - (\text{VectorA}[0] * \text{VectorB}[2])$$

$$\text{Vector}[2] = (\text{VectorA}[0] * \text{VectorB}[1]) - (\text{VectorA}[1] * \text{VectorB}[0])$$

cout << Vector[0] << ", " << [1] << ", " << [2];

Program Solution:

```

97 void CrossProduct() {
98
99     Vector[0] = (VectorA[1] * VectorB[2]) - (VectorA[2] * VectorB[1]);
100    Vector[1] = (VectorA[2] * VectorB[0]) - (VectorA[0] * VectorB[2]);
101    Vector[2] = (VectorA[0] * VectorB[1]) - (VectorA[1] * VectorB[0]);
102
103    cout << "Result: (" << Vector[0] << ", " << Vector[1] << ", " << Vector[2] << ")" << endl;
104
105    system("pause");
106
107 }
```

Testing:

Details of expected results, actual results and screenshots for all tests conducted can be located in appendix 5.

Test Number & Name	Description	Pass/Fail

3.7. Mean of a List of Integers

Code Solution Prior To Coding (Mean):

```

Mean - Buff code

for (int i = 0; i < dimensions; i++){
    Answer = Answer + Vector[i];
}

Answer = Answer / dimensions;

cout << Answer;

```

Program Solution (Mean):

```

109 void Mean() {
110     for (int i = 0; i < dimensions; i++)
111     {
112         Answer = Answer + Vector[i];
113     }
114     Answer = Answer / dimensions;
115     cout << "Result: " << Answer << endl;
116     system("pause");
117 }
118
119
120
121
122

```

Testing (Mean):

Details of expected results, actual results and screenshots for all tests conducted can be located in appendix 6.

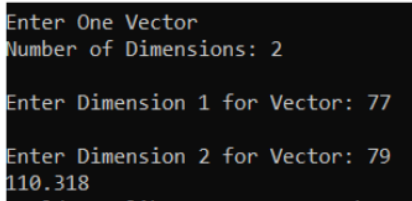
Test Number & Name	Description	Pass/Fail

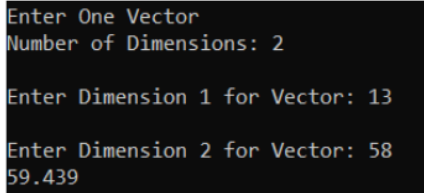
6. References

Reference the books on Moodle

7. Appendices

7.1. Appendix 1: Testing Schedule for Finding and returning the length of a Vector

Test Number & Name	1.1: 2D Vector
Description	To return the length of a vector, in this test we are checking for the correct return of the 2-dimensional vector of... $\vec{a} = (77 \ 79)$
Expected Result	The console should output a result of approximately = 110.3177
Actual Result	The console has the user select the number of dimensions. The console has the user then enter each dimension for the 2D vector and outputs the result to the console.  <pre> Enter One Vector Number of Dimensions: 2 Enter Dimension 1 for Vector: 77 Enter Dimension 2 for Vector: 79 110.318 </pre>
Pass/Fail	PASS

Test Number & Name	1.2: 2D Vector
Description	To return the length of a vector, in this test we are checking for the correct return of the 2-dimensional vector of... $\vec{a} = (13 \ 58)$
Expected Result	The console should output a result of approximately = 59.439
Actual Result	The console has the user select the number of dimensions. The console has the user then enter each dimension for the 2D vector and outputs the result to the console.  <pre> Enter One Vector Number of Dimensions: 2 Enter Dimension 1 for Vector: 13 Enter Dimension 2 for Vector: 58 59.439 </pre>
Pass/Fail	PASS

Test Number & Name	1.3: 2D Vector
Description	To return the length of a vector, in this test we are checking for the correct return of the 2-dimensional vector of... $\left \vec{a} \right = \left (89 \ 65) \right $
Expected Result	The console should output a result of approximately = 110.2089
Actual Result	The console has the user select the number of dimensions. The console has the user then enter each dimension for the 2D vector and outputs the result to the console. <pre>Enter One Vector Number of Dimensions: 2 Enter Dimension 1 for Vector: 89 Enter Dimension 2 for Vector: 65 110.209</pre>
Pass/Fail	PASS

Test Number & Name	2.1: 3D Vector
Description	To return the length of a vector, in this test we are checking for the correct return of the 3-dimensional vector of... $\left \vec{a} \right = \left (26 \ 24 \ 98) \right $
Expected Result	The console should output a result of approximately = 104.1921
Actual Result	The console has the user select the number of dimensions. The console has the user then enter each dimension for the 3D vector and outputs the result to the console. <pre>Enter One Vector Number of Dimensions: 3 Enter Dimension 1 for Vector: 26 Enter Dimension 2 for Vector: 24 Enter Dimension 3 for Vector: 98 104.192</pre>
Pass/Fail	PASS

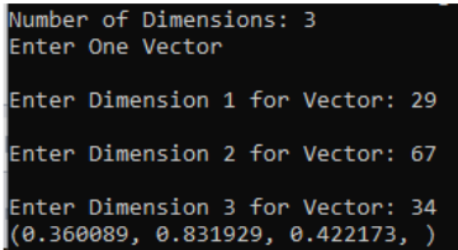
Test Number & Name	2.2: 3D Vector
Description	To return the length of a vector, in this test we are checking for the correct return of the 3-dimensional vector of... $\left \vec{a} \right = \left (87 \ 46 \ 76) \right $
Expected Result	The console should output a result of approximately = 124.3423
Actual Result	The console has the user select the number of dimensions. The console has the user then enter each dimension for the 3D vector and outputs the result to the console. <pre>Enter One Vector Number of Dimensions: 3 Enter Dimension 1 for Vector: 87 Enter Dimension 2 for Vector: 46 Enter Dimension 3 for Vector: 76 124.342</pre>
Pass/Fail	PASS

Test Number & Name	2.3: 3D Vector
Description	To return the length of a vector, in this test we are checking for the correct return of the 3-dimensional vector of... $\left \vec{a} \right = \left (29 \ 67 \ 34) \right $
Expected Result	The console should output a result of approximately = 80.54
Actual Result	The console has the user select the number of dimensions. The console has the user then enter each dimension for the 3D vector and outputs the result to the console. <pre>Enter One Vector Number of Dimensions: 3 Enter Dimension 1 for Vector: 29 Enter Dimension 2 for Vector: 67 Enter Dimension 3 for Vector: 34 80.5357</pre>
Pass/Fail	PASS

7.2. Appendix 2: Testing Schedule for Normalize a given vector

Test Number & Name	1.1: Test to Norm a 3D Vector
Description	To return a normalised vector, in this test we are checking for the correct normalisation of the 3-dimensional vector of... $\vec{a} = (26 \ 24 \ 98)$
Expected Result	The console should output a result of approximately = (0.24954, 0.23034, 0.9406)
Actual Result	The console has the user select the number of dimensions. The console has the user then enter each dimension for the 3D vector and outputs the result to the console. <pre>Enter One Vector Number of Dimensions: 3 Enter Dimension 1 for Vector: 26 Enter Dimension 2 for Vector: 24 Enter Dimension 3 for Vector: 98 (0.249539, 0.230344, 0.94057,)</pre>
Pass/Fail	PASS

Test Number & Name	1.2: Test to Norm a 3D Vector
Description	To return the length of a vector, in this test we are checking for the correct normalisation of the 3-dimensional vector of... $\vec{a} = (87 \ 46 \ 76)$
Expected Result	The console should output a result of approximately = (0.6997, 0.36995, 0.6112)
Actual Result	The console has the user select the number of dimensions. The console has the user then enter each dimension for the 3D vector and outputs the result to the console. <pre>Number of Dimensions: 3 Enter One Vector Enter Dimension 1 for Vector: 87 Enter Dimension 2 for Vector: 46 Enter Dimension 3 for Vector: 76 (0.699682, 0.369947, 0.611216,)</pre>
Pass/Fail	PASS

Test Number & Name	1.3: Test to Norm a 3D Vector
Description	To return the length of a vector, in this test we are checking for the correct normalisation of the 3-dimensional vector of... $\vec{a} = (29 \ 67 \ 34)$
Expected Result	The console should output a result of approximately = (0.3601, 0.832, 0.4222)
Actual Result	The console has the user select the number of dimensions. The console has the user then enter each dimension for the 3D vector and outputs the result to the console.  <pre> Number of Dimensions: 3 Enter One Vector Enter Dimension 1 for Vector: 29 Enter Dimension 2 for Vector: 67 Enter Dimension 3 for Vector: 34 (0.360089, 0.831929, 0.422173,) </pre>
Pass/Fail	PASS

7.3. Appendix 3: Testing Schedule for Finding the distance between two points

Test Number & Name	
Description	
Expected Result	
Actual Result	
Pass/Fail	

7.4. Appendix 4: Testing Schedule for calculating the dot product of two vectors

Test Number & Name	
Description	
Expected Result	
Actual Result	
Pass/Fail	

7.5. Appendix 5: Testing Schedule for calculating the cross product of two vectors

Test Number & Name	
Description	
Expected Result	
Actual Result	
Pass/Fail	

7.6. Appendix 6: Testing Schedule for calculating the Mean

Test Number & Name	
Description	
Expected Result	
Actual Result	
Pass/Fail	

7.7. Appendix 7: Code for Complete Program

```
#include <string>
#include <iostream>
#include <sstream>
#include <math.h>

using namespace std;

string input;
stringstream stream;
double VectorA[10000], VectorB[10000], Vector[10000];
int temp = 0;
double Answer = 0;
bool run = true;
int dimensions;

void Dimensions();
void Entries();

void TwoVectors();
void OneVector();
void OneSet();

void Distance();
void DotProduct();
void Length();
void Norm();
void CrossProduct();
void Mean();

int main(int argc, char** argv[]) {

    //Prints Main Menu

    do
    {
        system("cls");
        cout << "~~ Main Menu ~~" << endl;
        cout << "Please select from the following:" << endl << endl;
        cout << "1. Find and return the length of a Vector (Done) " << endl;
        cout << "2. Normalize a given vector (Done)" << endl;
        cout << "3. Find the distance between two points (Done)" << endl;
        cout << "4. The dot product of two vectors (Done)" << endl;
        cout << "5. The cross product of two vectors (Done)" << endl;
        cout << "6: Perform a rotation about the y axis transformation." <<
endl;

        cout << "7. Mean (Done), Mode, and Standard Deviation" << endl;
        cout << "8. The union and intersection of two sets" << endl;
        cout << "0. Exit" << endl << endl;
    }
}
```

```
        //Input for the user to select from the main menu

        cout << "Which option do you select? : ";
        getline(cin, input);
        stream.str(input);
        stream >> temp;
        stream.clear();

        //Switch statement to call the correct function depending on users
choice
        switch (temp)
        {
        case 1: { // Vector function calls

                cout << "You have choosen to find and return the length of a
vector." << endl << endl;

                Dimensions();
                OneVector();
                Length();

                break;
        }
        case 2: { // Normalize a given vector
                cout << "You have choosen to normalize a given vector." << endl;

                Dimensions();
                OneVector();
                Norm();

                break;
        }
        case 3: {
                cout << "You have choosen to calculate the distance between two
points." << endl;

                //Call function
                Dimensions();
                TwoVectors();
                Distance();

                break;
        }
        case 4: {

                cout << "You have choosen to calculate the dot product of two
vectors." << endl;

                //Call function
                Dimensions();
                TwoVectors();
                DotProduct();

                break;
        }
        case 5: {

                cout << "You have choosen to calculate the cross product of two
vectors." << endl;
```

```
        //Call
        Dimensions();
        TwoVectors();
        CrossProduct();

        break;
    }
    case 6: {

        cout << "You have choosen to perform a rotation about the y axis
transformation." << endl;

        //Call function

        break;
    }
    case 7: {

        cout << "You have choosen to calculate the Mean, Mode, and
Standard Deviation." << endl;

        //Call function

        Entries();
        OneSet();
        Mean();

        break;
    }
    case 8: {

        cout << "You have choosen to calculate the union and intersection
of two sets." << endl;

        //Call function

        break;
    }
    case 0: {
        run = false;
        break;
    }
    default:
        break;
    }
} while (run);

cout << endl << endl << "Thank you for using the program! ";
system("pause");
}

//User input for number of dimensions or entries
void Dimensions() {
    cout << "Number of Dimensions: ";
    getline(cin, input);
    stream.str(input);
    stream >> dimensions;
    stream.clear();
}
```

```
void Entries() {
    cout << "Number of Entries: ";
    getline(cin, input);
    stream.str(input);
    stream >> dimensions;
    stream.clear();
}

//User input for Vectors and Sets of Data

void TwoVectors() {

    cout << "Enter Two Vectors (A & B)" << endl;

    for (int i = 0; i < dimensions; i++)
    {
        temp = i + 1;
        cout << endl << "Enter Dimension " << temp << " for Vector A: ";
        getline(cin, input);
        stream.str(input);
        stream >> VectorA[i];
        stream.clear();
    }

    for (int i = 0; i < dimensions; i++)
    {
        temp = i + 1;
        cout << endl << "Enter Dimension " << temp << " for Vector B: ";
        getline(cin, input);
        stream.str(input);
        stream >> VectorB[i];
        stream.clear();
    }
}

void OneVector(){

    cout << "Enter One Vector" << endl;

    for (int i = 0; i < dimensions; i++)
    {
        temp = i + 1;
        cout << endl << "Enter Dimension " << temp << " for Vector: ";
        getline(cin, input);
        stream.str(input);
        stream >> Vector[i];
        stream.clear();
    }
}

void OneSet() {

    system("cls");

    cout << "Enter One Set of Data" << endl;

    for (int i = 0; i < dimensions; i++)
    {
        temp = i + 1;
        cout << endl << "Enter number " << temp << " : ";
        getline(cin, input);
    }
}
```

```
        stream.str(input);
        stream >> Vector[i];
        stream.clear();
    }

    for (int i = 0; i < dimensions; i++)
    {
        temp = i + 1;
        cout << "Entry " << temp << ": " << Vector[i] << endl;
    }

    system("pause");
}

//Calculation Functions

void Distance() {
    for (int i = 0; i < dimensions; i++)
    {
        Vector[i] = VectorA[i] - VectorB[i];
        Vector[i] = Vector[i] * Vector[i];
        Answer = Answer + Vector[i];
    }

    Answer = sqrt(Answer);

    cout << Answer;
}

void DotProduct() {
    for (int i = 0; i < dimensions; i++)
    {
        Vector[i] = VectorA[i] * VectorB[i];
        Answer = Answer + Vector[i];
    }

    cout << Answer;
}

void Length() {
    for (int i = 0; i < dimensions; i++)
    {
        Vector[i] = Vector[i] * Vector[i];
        Answer = Answer + Vector[i];
    }

    Answer = sqrt(Answer);

    cout << Answer;
}

void Norm() {
```

```
    for (int i = 0; i < dimensions; i++)
    {
        VectorA[i] = Vector[i] * Vector[i];
        Answer = Answer + VectorA[i];
    }

    Answer = sqrt(Answer);

    for (int i = 0; i < dimensions; i++)
    {
        Vector[i] = Vector[i] / Answer;
    }

    cout << "(";

    for (int i = 0; i < dimensions; i++)
    {
        cout << Vector[i] << ", ";
    }

    cout << ")" << endl;

    system("pause");
}

void CrossProduct() {

    Vector[0] = (VectorA[1] * VectorB[2]) - (VectorA[2] * VectorB[1]);
    Vector[1] = (VectorA[2] * VectorB[0]) - (VectorA[0] * VectorB[2]);
    Vector[2] = (VectorA[0] * VectorB[1]) - (VectorA[1] * VectorB[0]);

    cout << "Result : (" << Vector[0] << ", " << Vector[1] << ", " << Vector[2] <<
    ")" << endl;

    system("pause");
}

void Mean() {

    for (int i = 0; i < dimensions; i++)
    {
        Answer = Answer + Vector[i];
    }

    Answer = Answer / dimensions;

    cout << "Result (Mean): " << Answer << endl;

    system("pause");
}
```

FINAL GRADE

GENERAL COMMENTS

Instructor

43/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9



Comment 1

in general make sure your accumulator, here it is the variable Answer, is set to zero prior to entering the loop.

PAGE 10

PAGE 11

PAGE 12

PAGE 13



Comment 2

pass parameters don't use globals

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

GRADING FORM: SMU GENERIC_GJ

FATMA ABD AL KERIM

42.50

RESEARCH-INFORMED LITERATURE



You must include references, for example, as a minimum reference the course text. No all tasks attempted

2

KNOWLEDGE AND UNDERSTANDING

 For the tasks attempted the examples and testing and implementation was quite thorough. Well done here. Later sections missing unfortunately.

9

ANALYSIS

 Some later sections missing but calculations and methods used correct. Test design appropriate.

8.5

PRACTICAL APPLICATION AND DEP.

 For tasks attempted code in evidence. Avoid globals and try to pass by value or reference in your functions.

21

SKILLS FOR PROFESSIONAL PRAC.

 Some handwritten equations - have you tried Word's equation editor? Report structured. No refs. Some tasks missing.

2