

# Scott\_Howells\_1701604\_Assign ment

*by* Scott Howells

---

|                |                                                                               |                 |       |
|----------------|-------------------------------------------------------------------------------|-----------------|-------|
| FILE           | 4077_SCOTT_HOWELLS_SCOTT_HOWELLS_1701604_ASSIGNMENT_11<br>2721_314944284.DOCX |                 |       |
| TIME SUBMITTED | 25-APR-2018 05:13PM (UTC+0100)                                                | WORD COUNT      | 4207  |
| SUBMISSION ID  | 87458122                                                                      | CHARACTER COUNT | 19537 |

**Maths for Software Development:**

**Application of Maths for Software Development**

**Module Tutor: Gordon Dickers**

**By: Scott Howells**

**Student Number: 1701604**

## **Table of contents**

- 1. Introduction**
- 2. Length**
- 3. Normalize**
- 4. Distance between two Points**
- 5. Cross product of two vectors**
- 6. Multiply a (4x4) and (4x1) column matrix**
- 7. Mean**
- 8. Median**
- 9. Mode**
- 10. Standard Deviation**
- 11. Union and Intersection of two sets**
- 12. Testing's**
- 13. References**
- 14. Code Listings**

## 1. Introduction

The purpose of this report is to document the work constructed on a maths library for a new software application. This document will contain the descriptive information based on the actions that were required to be performed as part of the work conducted. The document will also contain the responsibilities for creating and testing the numerous functions that have been proposed by the supervisors. The report will also demonstrate these functions with appropriate testing and examples.

The code that was constructed was completed within Visual Studio. In order for the program to come across as user friendly [1], the use of a menu was developed in order for the user to choose what functions they wish to perform. Below is a screenshot of the menu that was constructed:

```
1 - Length.
2 - Normalize.
3 - Distance between two Points.
4 - Cross product of two vectors.
5 - Multiply a (4x4) and (4x1) column matrix.
6 - Mean of a list of integers.
7 - Median of a list of integers.
8 - Mode of a list of integers.
9 - Standard Deviation of a list of integers.
10 - Union and Intersection of two sets.
Please enter an option from the menu!
```

```
void length() { ... }
void normalize() { ... }
void distance() { ... }
void crossProduct() { ... }
void matrix() { ... }
void mean() { ... }
void median() { ... }
void mode() { ... }
void standardDeviation() { ... }
```

The menu was constructed by using voids in order to call upon:

```
switch (userOption)
{
case 1:
    length();//code to launch Length
    break;
case 2:
    normalize();//code to launch Normalize
    break;
case 3:
    distance();//code to make launch Distance between two points
    break;
case 4:
    crossProduct();//code to launch Cross product of two vectors
    break;
case 5:
    matrix();//code to launch Multiply a (4x4) and (4x1) column matrix
    break;
case 6:
```

If a user enters an invalid option from the menu then an error message will be displayed:

```
10 - Union and Intersection of two sets.
Please enter an option from the menu!
11
Not a Valid Choice!
Choose again!
```

## 2. Length

This function finds and returns the length of a vector. The length of a vector can be measured by using its magnitude (the magnitude of a vector is the length of a vector). In order to find the length of a vector the following equations must be performed:

Length = square root  $((ax * ax) + (ay * ay) + (az * az))$

For example, within the code that was produced, the following coordinates were used in order to portray the components:

```
ax = 6;  
ay = 4;  
az = 5;
```

To calculate the magnitude (length) of the vector, all that is then needed is the following calculation:

```
param = (ax * ax) + (ay * ay) + (az * az);  
length = sqrt(param);
```

```
void length()  
{  
    double ax = 6;  
    double ay = 4;  
    double az = 5;  
    double param = 0;  
    double length;  
  
    param = (ax * ax) + (ay * ay) + (az * az);  
    length = sqrt(param);  
  
    printf("sqrt(%f) = %f\n", param, length);  
}
```

(The code demonstrated above shows how the program will be executed)

The above screenshot demonstrates how the length of a vector will be found and displayed. The components are entered as ax = 6, ay = 4 and az = 5. The calculations are performed and then displayed for the user to see. Below is the screenshot of the code being executed:

```
1 - Length.  
2 - Normalize.  
3 - Distance between two Points.  
4 - Cross product of two vectors.  
5 - Multiply a (4x4) and (4x1) column matrix.  
6 - Mean of a list of integers.  
7 - Median of a list of integers.  
8 - Mode of a list of integers.  
9 - Standard Deviation of a list of integers.  
10 - Union and Intersection of two sets.  
Please enter an option from the menu!  
1  
sqrt(77.000000) = 8.774964
```

By using OnlineMSchool [2], the following equation proves successful as the correct value has been displayed.

### 3. Normalize

This function normalizes the given function. The expression to normalize a vector is to take the length given to a vector and change its length to 1, thus turning into what is called a unit vector. In order to achieve this the length must be calculated then, divide each of its components by its length.

For example, within the code that was produced the components consist of:

```
ax = 6;  
ay = 4;  
az = 5;
```

To normalize these given components, the calculations are as followed:

$$ax = 6.0 / |a|$$
$$ay = 4.0 / |a|$$
$$az = 5.0 / |a|$$

As figured out within the previous functions, the length of the vector is 8.774. This figure can be placed where the ' $|a|$ ' has been implemented. The results are as followed:

$$ax = 6.0 / 8.774 = 0.683$$
$$ay = 4.0 / 8.774 = 0.455$$
$$az = 5.0 / 8.774 = 0.569$$

The code that was constructed has been issued below:

```
void normalize()  
{  
    double ax = 6.0;  
    double ay = 4.0;  
    double az = 5.0;  
    double param = 8.774;  
    double normalize = 0;  
  
    normalize = (ax / param);  
    printf("normalized component of ax is:\r\n %f\r\n", normalize);  
    normalize = (ay / param);  
    printf("normalized component of ay is:\r\n %f\r\n", normalize);  
    normalize = (az / param);  
    printf("normalized component of az is:\r\n %f\r\n", normalize);  
}
```

The code takes the values of the components ax, ay and az and divides them by the parameter set (length of the vector).

```
1 - Length.  
2 - Normalize.  
3 - Distance between two Points.  
4 - Cross product of two vectors.  
5 - Multiply a (4x4) and (4x1) column matrix.  
6 - Mean of a list of integers.  
7 - Median of a list of integers.  
8 - Mode of a list of integers.  
9 - Standard Deviation of a list of integers.  
10 - Union and Intersection of two sets.  
Please enter an option from the menu!  
2  
normalized component of ax is:  
0.683839  
normalized component of ay is:  
0.455892  
normalized component of az is:  
0.569866
```

The executed code correctly displays the correct value for each of the components that have been issued.

#### 4. Distance between two Points

This function takes two points and returns the distance between them. The distance between two points generally means the shortest distance. For example, there are many different amounts of paths between two points on a sphere however, in general, only a single shortest path. The distance formula is an algebraic expression used to determine the distance within the coordinates (x1, y1) and (x2, y2).



```
void distance()
{
    double x = 0;
    double y = 0;
    double a = 0;
    double b = 0;
    double answer = 0;

    double distanceBetweenTwoPoints(double x, double y, double a, double b);

    printf("Enter the points for the x coordinates\n");
    scanf_s("%lf", &x);
    printf("Enter the points for the y coordinates\n");
    scanf_s("%lf", &y);
    printf("Enter the points for the second x coordinate\n");
    scanf_s("%lf", &a);
    printf("Enter the points for the second y coordinate\n");
    scanf_s("%lf", &b);
    answer = sqrt(pow(x - a, 2) + pow(y - b, 2));
    printf("The answer is:\n %0.2f\n", answer);
}
```

The code above demonstrates the calculations performed in order to display the correct result. By using a double variable, the user can enter numbers that contain a decimal point. The code allows the user to enter their own coordinates. The answer will only be correct if the formula  $\sqrt{(x - a)^2 + (y - b)^2}$  is issued.

```
5 - Multiply a (4x4) and (4x1) column matrix.
6 - Mean of a list of integers.
7 - Median of a list of integers.
8 - Mode of a list of integers.
9 - Standard Deviation of a list of integers.
10 - Union and Intersection of two sets.
Please enter an option from the menu!
3
Enter the points for the x coordinates
2
Enter the points for the y coordinates
3
Enter the points for the second x coordinate
3
Enter the points for the second y coordinate
2
The answer is:
1.41
```



The following screenshot proves that the correct values have been displayed.

## 5. Cross Product of two Vectors

This function calculates the dot product of two vectors and returns the result. The dot product is the sum of the products of the corresponding entries of the two sequences of numbers. The dot product of two vectors will be equal to the cosine of the angle between the vectors, times the lengths of each of the vectors. For example, if A and B are perpendicular (at 90 degrees to one another), the result of the dot product will be zero, because cos will be zero.

```
void crossProduct()
{
    double a0 = 0;
    double a1 = 0;
    double a2 = 0;
    double b0 = 0;
    double b1 = 0;
    double b2 = 0;

    double A0 = 0;
    double A1 = 0;
    double A2 = 0;

    double twoDTotal, threeDTotal;

    printf("Please enter the value for a1:\n");
    scanf_s("%lf", &a0);
    printf("Please enter the value for a2:\n");
    scanf_s("%lf", &a1);
    printf("Please enter the value for a3:\n");
    scanf_s("%lf", &a2);

    printf("Please enter the value for b1:\n");
    scanf_s("%lf", &b0);
    printf("Please enter the value for b2:\n");
    scanf_s("%lf", &b1);
    printf("Please enter the value for b3:\n");
    scanf_s("%lf", &b2);

    twoDTotal = ((a0 * b0) + (a1 * b1));
    threeDTotal = ((a0 * b0) + (a1 * b1) + (a2 * b2));

    printf("The 2D Total is:\n\n%0.2f\n", twoDTotal);
    printf("The 3D Total is:\n\n%0.2f\n", threeDTotal);
}
```

The code that was constructed allows for the user to enter the values they wish to use. The code allows for the inclusion of a 2D total and a 3D total.

The calculations for the 2D total is:

$$\text{twoDTotal} = ((a0 * b0) + (a1 * b1));$$

The calculations for the 3D total is:

$$\text{threeDTotal} = ((a0 * b0) + (a1 * b1) + (a2 * b2));$$

In order to display the correct 2D total, the value entered on the third value prompt (a3) must either be 0 or 1 to correctly display.

```
Please enter an option from the menu!
4
Please enter the value for a1:
4
Please enter the value for a2:
3
Please enter the value for a3:
5
Please enter the value for b1:
2
Please enter the value for b2:
3
Please enter the value for b3:
4
The 2D Total is:
17.00
The 3D Total is:
37.00
```

As shown, the code correctly displays the correct value when the user enters the separate values.

## 6. Multiply a (4x4) and (4x1) column matrix

This function takes 2 matrices of the given order, multiplies them and returns the resulting matrix. In order to multiply two matrices, the number of columns within the first matrix should always be equal to the number of rows within the second matrix. The program constructed will display the error until the number has correctly been changed.

```
printf("Error! column of first matrix not equal to row of second.");

printf("Enter rows for first matrix:\r\n");
scanf_s("%i", &r1);
printf("Enter columns for first matrix:\r\n");
scanf_s("%i", &c1);

printf("Enter rows for second matrix:\r\n");
scanf_s("%i", &r2);
printf("Enter columns for second matrix:\r\n");
scanf_s("%i", &c2);
}

// Storing elements of first matrix.
printf("Enter elements of matrix 1:\r\n");
for (i = 0; i < r1; ++i)
    for (j = 0; j < c1; ++j)
    {
        printf("Enter element a\r\n", i + 1 << j + 1);
        scanf_s("%i", &a[i][j]);
    }

// Storing elements of second matrix.
printf("Enter elements of matrix 2:\r\n");
for (i = 0; i < r2; ++i)
    for (j = 0; j < c2; ++j)
    {
        printf("Enter element b\r\n", i + 1 << j + 1);
```

(this is a snippet of the code, the full code listings have been made available within the code section)

```
Enter rows for first matrix:
2
Enter columns for first matrix:
2
Enter rows for second matrix:
2
Enter columns for second matrix:
2
Enter elements of matrix 1:
Enter element a 4
Enter element a
4
Enter element a
5
Enter element a
6
Enter elements of matrix 2:
Enter element b
3
Enter element b
2
Enter element b
7
Enter element b
9
Output Matrix:
37
42
57
64
```

The code displays the matrix of the numbers that the user has entered.

```
Enter rows for first matrix:
2
Enter columns for first matrix:
3
Enter rows for second matrix:
2
Enter columns for second matrix:
2
Error! column of first matrix not equal to row of second.Enter rows for first matrix:
```

The error correctly is displayed when the wrong values have been entered.

## 7. Mean

The Mean of a set of numbers is the average. In order to find the average of a set of numbers, you must first add them up and divide by the number of numbers there are.

```
void mean()
{
    int data;
    int i;
    double num[100];
    double sum = 0.0;
    double mean;

    printf("Enter total numbers of data:\r\n");
    scanf("%i", &data);

    for (i = 0; i < data; i++)
    {
        printf("Enter number:\r\n", i + 1);
        scanf("%lf", &num[i]);
        sum = sum + num[i];
    }
    mean = sum / data;
    printf("The Mean is:\r\n %.2f\r\n", mean);
}
```

The code above allows the user to enter the amount of numbers they wish to add. The for loop will add the numbers together, whilst the mean will then be calculated by taking the sum of the addition and dividing by how many numbers the user has entered.

```
Please enter an option from the menu!
6
Enter total numbers of data:
3
Enter number:
2
Enter number:
3
Enter number:
2
The Mean is:
2.33
```

Unfortunately, the displayed number for the Mean is incorrect thus proving an error within the code. The correct value should display 5.66. Despite this the code works to an extent, however, more time to fix this code is needed.

## 8. Median

The median is the number that is halfway within the set of numbers that are on offer. To find the median, the data should be arranged in order from the least to the greatest. If there is an even number of numbers within the set, then the median is found by taking the mean of the two middle numbers.

```
void median()
{
    float median[] = { 10, 25, 19, 20, 27, 30, 21, 8, 18, 16 }; //Set of numbers
    int x;
    int y;
    float temp;
    int size = 11;

    for (x = 0; x < size; x++)
    {
        for (y = x + 1; y < size; y++)
        {
            if (median[y] > median[x])
            {
                temp = median[x];
                median[x] = median[y];
                median[y] = temp;
            }
        }
    }

    if (size % 2 == 0)
        printf("The median is\n%f\n", (median[size / 2] + median[size / 2 - 1]) / 2); //If even
    else
        printf("The median is\n%f\n", median[size / 2]);
}
```

It was difficult to allow the user to enter the numbers they wish and to find the median from those sets (which is why the mean might not have worked correctly). The implementation of a set of numbers were displayed within the code as floats in order to calculate the floating decimal points. The code was also constructed to keep in mind that the possibility of 2 numbers can be chosen.

```
1 - Length.
2 - Normalize.
3 - Distance between two Points.
4 - Cross product of two vectors.
5 - Multiply a (4x4) and (4x1) column matrix.
6 - Mean of a list of integers.
7 - Median of a list of integers.
8 - Mode of a list of integers.
9 - Standard Deviation of a list of integers.
10 - Union and Intersection of two sets.
Please enter an option from the menu!
7
The median is
19.000000
```

The correct value has been chosen as the number 19 within the set was the middle number when order from lowest to the highest.

## 9. Mode

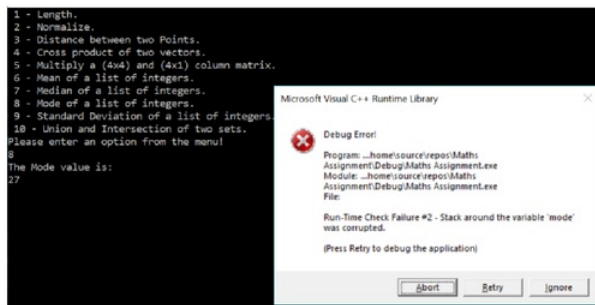
The mode of a set of numbers is the number that occurs most frequently in the set. The number that occurs the most within a set of numbers is the mode.

```
void mode()
{
    int i;
    int j;
    int z;
    int temp;
    int maxCount;
    int modeValue;
    float mode[] = { 10, 25, 19, 20, 27, 30, 20, 8, 18, 16 };
    int size = 11;
    int tally[11];

    for (i = 0; i < size; i++)
    {
        for (j = 0; j < size - i; j++)
        {
            if (mode[j] > mode[j + 1])
            {
                temp = mode[j];
                mode[j] = mode[j + 1];
                mode[j + 1] = temp;
            }
        }
    }
    for (i = 0; i < size; i++)
```

(this is a snippet of the code, the full code listings have been made available within the code section)

This proved extremely difficult as the code from the median was also implemented within the mode code in order to garner the correct and simplest way in order to produce the code required.



Unfortunately, an error message keeps being displayed when the mode has been selected.

## 10. Standard Deviation

Standard Deviation is a measure of how spread out numbers are. A low Standard Deviation means that most of the numbers are very close to the average. A high Standard Deviation means that the numbers are spread out. In order to calculate the Standard Deviation, the mean must be calculated. Then each number must subtract the mean and square the results. The mean of those squared differences must be figured out before taking the square root.

```
void standardDeviation()
{
    int number_of_items;
    printf("Enter total numbers of data:\n\n");
    scanf_s("%i", &number_of_items);

    float list[11];

    // get all the data items
    printf("Please enter the data:\n\n");
    int i;
    for (i = 1; i <= number_of_items; i++)
    {
        printf(" ", i);
        scanf_s("%f", &list[i]);
    }

    // find the mean (average) of the data
    float xbar = 0;
    for (i = 0; i <= number_of_items; i++)
        xbar = xbar + list[i];

    // find the standard deviation
    float numerator = 0;
    float denominator = number_of_items;

    for (i = 1; i <= number_of_items; i++)
        numerator = numerator + pow((list[i] - xbar), 2);

    float standard_deviation = sqrt(numerator / denominator);
    printf("The standard deviation for the given data is:\n\n%f\n\n", standard_deviation);
    scanf_s("%f", &standard_deviation);
}
```

```
1 - Length.
2 - Normalize.
3 - Distance between two Points.
4 - Cross product of two vectors.
5 - Multiply a (4x4) and (4x1) column matrix.
6 - Mean of a list of integers.
7 - Median of a list of integers.
8 - Mode of a list of integers.
9 - Standard Deviation of a list of integers.
10 - Union and Intersection of two sets.
Please enter an option from the menu!
9
Enter total numbers of data:
5
Please enter the data:
40
20
30
5
2
The standard deviation for the given data is:
107374096.000000
```

Unfortunately, not all of the numbers that the users enter will work. There is an error within the code that doesn't allow for the correct amount to be displayed.

## 11. Union and Intersection of two sets

The Union of two sets is the elements which are in either set, whereas the Intersection of two sets is the set of elements which are in both sets. An example of this would be A = (1,2,3) and B = (3,4,5). The Union will display AB = (1,2,3,4,5) and the intersection will display AB = (3).

```
Assignment
for (; i < n; ++i)
{
    u[k] = a1[i];
    k++;
}
else if (j < m)
{
    for (; j < m; ++j)
    {
        u[k] = a2[j];
        k++;
    }
}

printf("Union of two arrays is:\r\n");
for (i = 0; i < k; ++i)
{
    printf("%d\r\n", u[i]);
}

printf(" The Intersection is:\r\n");
for (i = 0, j = 0; i < 5 && j < 5;)
{
    if (a1[i] < a2[j])
        ++i;
    else if (a1[i] > a2[j])
        ++j;
    else
    {
        printf("%3d\r\n", a1[i]);
        ++i;
    }
}
```

The user enters the amount they would like to be placed within the first array. The user must then choose a set of numbers, the same goes for the second array. The union will place these numbers within ascending order from lowest to highest. The intersection will record if any number has been duplicated (used more than once) and display what that number or numbers are.

(this is a snippet of the code, the full code listings have been made available within the code section)

```
Enter size of first array:
3
Enter elements of first array in ascending order:
4
6
8
Enter size of second array:
4
Enter elements of second array in ascending order:
3
4
5
7
Union of two arrays is:
3
4
5
6
7
8
The intersection is:
4
```

## 12. Testing's

| Test Number: | Test Name:                                | Description:                                                                                                                                       | Expected Result:                                                                                                              | Actual Result:                                                                                                                 |
|--------------|-------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| 1.           | Length                                    | When length in the option menu has been chosen, the correct value will be displayed.                                                               | 8.774964 will be displayed.                                                                                                   | 8.774964 was displayed.                                                                                                        |
| 2.           | Normalize                                 | When normalizing, the components of the vector previously conducted must be divided by the length of the vector that had been acquired previously. | The correct values to be displayed:<br>$ax = 6.0 / 8.774 = 0.683$<br>$ay = 4.0 / 8.774 = 0.455$<br>$az = 5.0 / 8.774 = 0.569$ | The values were correctly displayed:<br>$ax = 6.0 / 8.774 = 0.683$<br>$ay = 4.0 / 8.774 = 0.455$<br>$az = 5.0 / 8.774 = 0.569$ |
| 3.           | Distance between two points               | Takes two points and returns the distance between them.                                                                                            | The correct value to be displayed when user enters number.                                                                    | The correct values were displayed with whatever number the user entered.                                                       |
| 4.           | Cross product of two vectors              | This function calculates the dot product of two vectors and returns the result                                                                     | The 2D and 3D total to correctly be displayed with whatever the user enters.                                                  | The 2D and 3D total were correctly displayed.                                                                                  |
| 5.           | Multiply a (4x4) and (4x1) column matrix. | This function takes 2 matrices of the given order, multiplies them and returns the resulting matrix.                                               | The value to be displayed correctly as a matrix. The error message to appear when wrong value entered.                        | The numbers were displayed within the matrix. The error message was also displayed.                                            |
| 6.           | Mean                                      | Add the amount of numbers the user enters and divide them by how have been implemented.                                                            | Collect the numbers and display the correct value.                                                                            | The numbers were collected however, the correct value has not displayed correctly.                                             |

|     |                                    |                                                                                                                                                                                                                                                                             |                                                                                     |                                                                                                                   |
|-----|------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|
| 7.  | Median                             | Sort the set of numbers from lowest to highest and select the middle number.                                                                                                                                                                                                | The number 19 to be displayed as the middle number.                                 | Number 19 was correctly displayed as the number.                                                                  |
| 8.  | Mode                               | The number that occurs the most often within the number set.                                                                                                                                                                                                                | The number 20 should be displayed as the most number to occur.                      | An error message is displayed rather than the number 20.                                                          |
| 9.  | Standard Deviation                 | Standard Deviation is a measure of how spread out numbers are                                                                                                                                                                                                               | The correct Standard Deviation to be displayed.                                     | Only a very few numbers correctly are displayed, otherwise all numbers are incorrect.                             |
| 10. | Union and Intersection of two sets | The functions take two arrays that represent the members of two sets. The Union function returns an array that represents the set that is the union of both sets. The Intersection function returns an array that represents the set that is the intersection of both sets. | To display the correct union output.<br>To display the correct Intersection output. | The correct display was shown for the union output.<br>The correct display was shown for the Intersection output. |

## 13. References

- [1] V. Hogg, "The importance of User-Friendly Website Design," imsmarketing, 27 January 2017. [Online]. Available: <https://www.imsmarketing.ie/design-services/the-importance-of-user-friendly-website-design/>. [Accessed 11 April 2018].
- [2] OnlineMSchool, [Online]. Available: <http://onlinemschool.com/math/assistance/vector/length/>. [Accessed 11 April 2018].

## 14. Code Listings

```
15. #define _CRT_SECURE_NO_WARNINGS
16. #include <string>
17. #include <math.h>
18.
19. using namespace std;
20.
21. void length()
22. {
23.     double ax = 6; //components
24.     double ay = 4; //components
25.     double az = 5; //components
26.     double param = 0; //parameter
27.     double length; //final length
28.
29.     param = (ax * ax) + (ay * ay) + (az * az); //calculation
30.     length = sqrt(param);
31.
32.     printf("sqrt(%f) = %f\n", param, length); //square root calculation and
    display
33.
34. }
35.
36. void normalize()
37. {
38.     double ax = 6.0; //Component
39.     double ay = 4.0; //Component
40.     double az = 5.0; //Component
41.     double param = 8.774; //Length of Vector
42.     double normalize = 0;
43.
44.     normalize = (ax / param);
45.     printf("normalized component of ax is:\r\n %f\n", normalize);
46.     normalize = (ay / param);
47.     printf("normalized component of ay is:\r\n %f\n", normalize);
48.     normalize = (az / param);
49.     printf("normalized component of az is:\r\n %f\n", normalize);
50. }
51.
52. void distance()
53. {
54.     double x = 0; //User enters
55.     double y = 0; //User enters
56.     double a = 0; //User enters
57.     double b = 0; //User enters
58.     double answer = 0;
59.
60.     double distanceBetweenTwoPoints(double x, double y, double a, double b);
61.
62.     printf("Enter the points for the x coordinates\r\n");
63.     scanf_s("%lf", &x);
64.     printf("Enter the points for the y coordinates\r\n");
65.     scanf_s("%lf", &y);
66.     printf("Enter the points for the second x coordinate\r\n");
67.     scanf_s("%lf", &a);
68.     printf("Enter the points for the second y coordinate\r\n");
69.     scanf_s("%lf", &b);
70.     answer = sqrt(pow(x - a, 2) + pow(y - b, 2)); //Square root calculation
    for answer
71.     printf("The answer is:\r\n %0.2f\r\n", answer);
72.
73.
```

```

74. }
75.
76. void crossProduct()
77. {
78.     double a0 = 0; //User enters
79.     double a1 = 0; //User enters
80.     double a2 = 0; //User enters
81.     double b0 = 0; //User enters
82.     double b1 = 0; //User enters
83.     double b2 = 0; //User enters
84.
85.     double A0 = 0;
86.     double A1 = 0;
87.     double A2 = 0;
88.
89.     double twoDTotal, threeDTotal;
90.
91.     printf("Please enter the value for a1:\r\n");
92.     scanf_s("%lf", &a0);
93.     printf("Please enter the value for a2:\r\n");
94.     scanf_s("%lf", &a1);
95.     printf("Please enter the value for a3:\r\n");
96.     scanf_s("%lf", &a2);
97.
98.     printf("Please enter the value for b1:\r\n");
99.     scanf_s("%lf", &b0);
100.    printf("Please enter the value for b2:\r\n");
101.    scanf_s("%lf", &b1);
102.    printf("Please enter the value for b3:\r\n");
103.    scanf_s("%lf", &b2);
104.
105.    twoDTotal = ((a0 * b0) + (a1 * b1)); //2D calculation
106.    threeDTotal = ((a0 * b0) + (a1 * b1) + (a2 * b2)); //3D
    Calculation
107.
108.    printf("The 2D Total is:\r\n %0.2f\r\n", twoDTotal); //Display 2D
    Total
109.    printf("The 3D Total is:\r\n %0.2f\r\n", threeDTotal); //Display
    3D Total
110.
111. }
112.
113. void matrix()
114. {
115.     int a[10][10];
116.     int b[10][10];
117.     int mult[10][10];
118.     int r1 = 0;
119.     int c1 = 0;
120.     int r2 = 0;
121.     int c2 = 0;
122.     int i = 0;
123.     int j = 0;
124.     int k = 0;
125.
126.     printf("Enter rows for first matrix:\r\n");
127.     scanf_s("%i", &r1);
128.     printf("Enter columns for first matrix:\r\n");
129.     scanf_s("%i", &c1);
130.     printf("Enter rows for second matrix:\r\n");
131.     scanf_s("%i", &r2);
132.     printf("Enter columns for second matrix:\r\n");

```

```

133.         scanf_s("%i", &c2);
134.
135.         // If column of first matrix is not equal to row of second
matrix,
136.         // ask the user to enter the size of matrix again.
137.         while (c1 != r2)
138.         {
139.             printf("Error! column of first matrix not equal to row of
second.");
140.
141.             printf("Enter rows for first matrix:\r\n");
142.             scanf_s("%i", &r1);
143.             printf("Enter columns for first matrix:\r\n");
144.             scanf_s("%i", &c1);
145.
146.             printf("Enter rows for second matrix:\r\n");
147.             scanf_s("%i", &r2);
148.             printf("Enter columns for second matrix:\r\n");
149.             scanf_s("%i", &c2);
150.         }
151.
152.         // Storing elements of first matrix.
153.         printf("Enter elements of matrix 1:\r\n");
154.         for (i = 0; i < r1; ++i)
155.             for (j = 0; j < c1; ++j)
156.             {
157.                 printf("Enter element a\r\n", i + 1 << j + 1);
158.                 scanf_s("%i", &a[i][j]);
159.             }
160.
161.         // Storing elements of second matrix.
162.         printf("Enter elements of matrix 2:\r\n");
163.         for (i = 0; i < r2; ++i)
164.             for (j = 0; j < c2; ++j)
165.             {
166.                 printf("Enter element b\r\n", i + 1 << j + 1);
167.                 scanf_s("%i", &b[i][j]);
168.             }
169.
170.         // Initializing elements of matrix mult to 0.
171.         for (i = 0; i < r1; ++i)
172.             for (j = 0; j < c2; ++j)
173.             {
174.                 mult[i][j] = 0;
175.             }
176.
177.         // Multiplying matrix a and b and storing in array mult.
178.         for (i = 0; i < r1; ++i)
179.             for (j = 0; j < c2; ++j)
180.                 for (k = 0; k < c1; ++k)
181.                 {
182.                     mult[i][j] += a[i][k] * b[k][j];
183.                 }
184.
185.         // Displaying the multiplication of two matrix.
186.         printf("Output Matrix:\r\n");
187.         for (i = 0; i < r1; ++i)
188.             for (j = 0; j < c2; ++j)
189.             {
190.                 printf("%i\r\n", mult[i][j]);
191.                 if (j == c2 - 1)
192.                     break;
193.             }

```

```

193.     }
194.
195.     void mean()
196.     {
197.         int data; //Numbers entered by user
198.         int i;
199.         double num[100];
200.         double sum = 0.0; //Value
201.         double mean;
202.
203.         printf("Enter total numbers of data:\r\n");
204.         scanf_s("%i", &data);
205.
206.         for (i = 0; i < data; i++)
207.         {
208.             printf("Enter number:\r\n", i + 1);
209.             scanf_s("%lf", &num[i]);
210.             sum = sum + num[i]; //Adding
211.         }
212.         mean = sum / data; //Mean calculation
213.         printf("The Mean is:\r\n %0.2f\r\n", mean);
214.
215.     }
216.
217.     void median()
218.     {
219.         float median[] = { 10, 25, 19, 20, 27, 30, 21, 8, 18, 16 }; //Set
of numbers
220.         int x;
221.         int y;
222.         float temp;
223.         int size = 11;
224.
225.
226.         for (x = 0; x < size; x++)
227.         {
228.             for (y = x + 1; y < size; y++)
229.             {
230.                 if (median[y] > median[x])
231.                 {
232.                     temp = median[x];
233.                     median[x] = median[y];
234.                     median[y] = temp;
235.                 }
236.             }
237.         }
238.
239.         if (size % 2 == 0)
240.             printf("The median is\r\n%f\r\n", (median[size / 2] +
median[size / 2 - 1]) / 2); //If even
241.         else
242.             printf("The median is\r\n%f\r\n", median[size / 2]);
243.     }
244.
245.     void mode()
246.     {
247.         int i;
248.         int j;
249.         int z;
250.         int temp;
251.         int maxCount;
252.         int modeValue;

```

```

253.         float mode[] = { 10, 25, 19, 20, 27, 30, 20, 8, 18, 16 }; //Set
of numbers
254.         int size = 11;
255.         int tally[11];
256.
257.         for (i = 0; i<size; i++)
258.         {
259.             for (j = 0; j<size - i; j++)
260.             {
261.                 if (mode[j]>mode[j + 1])
262.                 {
263.                     temp = mode[j];
264.                     mode[j] = mode[j + 1];
265.                     mode[j + 1] = temp;
266.                 }
267.             }
268.         }
269.         for (i = 0; i < size; i++)
270.         {
271.             for (z = i + 1; z<size; z++)
272.             {
273.                 if (mode[i] == mode[z])
274.                 {
275.                     tally[i]++;
276.                 }
277.             }
278.         }
279.         maxCount = 0;
280.         modeValue = 0;
281.         for (i = 0; i < size; i++)
282.         {
283.             if (mode[i] > maxCount)
284.             {
285.                 maxCount = tally[i];
286.                 modeValue = mode[i];
287.             }
288.         }
289.         printf("The Mode value is:\r\n%i\r\n", modeValue);
290.     }
291.
292.     void standardDeviation()
293.     {
294.         int number_of_items;
295.         printf("Enter total numbers of data:\r\n");
296.         scanf_s("%i", &number_of_items);
297.
298.         float list[11];
299.
300.         // get all the data items
301.         printf("Please enter the data:\r\n");
302.         int i;
303.         for (i = 1; i <= number_of_items; i++)
304.         {
305.             printf(" ", i);
306.             scanf_s("%f", &list[i]);
307.         }
308.
309.         // find the mean (average) of the data
310.         float xbar = 0;
311.         for (i = 0; i <= number_of_items; i++)
312.             xbar = xbar + list[i];
313.

```

```

314.         // find the standard deviation
315.         float numerator = 0;
316.         float denominator = number_of_items;
317.
318.         for (i = 1; i <= number_of_items; i++)
319.             numerator = numerator + pow((list[i] - xbar), 2);
320.
321.         float standard_deviation = sqrt(numerator / denominator);
322.         printf("The standard deviation for the given data is:\r\n%f\r\n",
standard_deviation);
323.         scanf_s("%f", &standard_deviation);
324.     }
325.
326.     void unionIntersection()
327.     {
328.         int a1[20];
329.         int a2[20];
330.         int u[40];
331.         int i;
332.         int j;
333.         int k;
334.         int n;
335.         int m;
336.         int lengthA = 0;
337.         int lengthB = 0;
338.
339.         printf("Enter size of first array:\r\n");
340.         scanf_s("%d", &n);
341.         printf("Enter elements of first array in ascending order:\r\n");
342.         for (i = 0; i < n; ++i)
343.         {
344.             scanf_s("%d", &a1[i]);
345.         }
346.
347.         printf("Enter size of second array:\r\n");
348.         scanf_s("%d", &m);
349.         printf("Enter elements of second array in ascending order:\r\n");
350.         for (i = 0; i < m; ++i)
351.         {
352.             scanf_s("%d", &a2[i]);
353.         }
354.
355.         for (i = 0, j = 0, k = 0; i < n && j < m;)
356.         {
357.             if (a1[i] < a2[j])
358.             {
359.                 u[k] = a1[i];
360.                 i++;
361.                 k++;
362.             }
363.             else if (a1[i] > a2[j])
364.             {
365.                 u[k] = a2[j];
366.                 j++;
367.                 k++;
368.             }
369.             else
370.             {
371.                 u[k] = a1[i];
372.                 i++;
373.                 j++;
374.                 k++;

```

```

375.         }
376.     }
377.
378.     if (i<n)
379.     {
380.         for (; i<n; ++i)
381.         {
382.             u[k] = a1[i];
383.             k++;
384.         }
385.     }
386.     else if (j<m)
387.     {
388.         for (; j<m; ++j)
389.         {
390.             u[k] = a2[j];
391.             k++;
392.         }
393.     }
394.
395.     printf("Union of two arrays is:\r\n");
396.     for (i = 0; i<k; ++i)
397.     {
398.         printf("%d\r\n", u[i]);
399.     }
400.
401.     printf(" The intersection is:\r\n");
402.     for (i = 0, j = 0; i < 5 && j < 5;)
403.     {
404.         if (a1[i] < a2[j])
405.             ++i;
406.         else if (a1[i] > a2[j])
407.             ++j;
408.         else
409.         {
410.             printf("%3d\r\n", a1[i]);
411.             ++i;
412.         }
413.     }
414. }
415.
416. int main(int argc, char** argv)
417. {
418.     int choice;
419.     int userOption = 0;
420.
421.     do
422.     {
423.         printf(" 1 - Length.\r\n");
424.         printf(" 2 - Normalize.\r\n");
425.         printf(" 3 - Distance between two Points.\r\n");
426.         printf(" 4 - Cross product of two vectors.\r\n");
427.         printf(" 5 - Multiply a (4x4) and (4x1) column
matrix.\r\n");
428.         printf(" 6 - Mean of a list of integers.\r\n");
429.         printf(" 7 - Median of a list of integers.\r\n");
430.         printf(" 8 - Mode of a list of integers.\r\n");
431.         printf(" 9 - Standard Deviation of a list of
integers.\r\n");
432.         printf(" 10 - Union and Intersection of two sets.\r\n");
433.
434.         printf("Please enter an option from the menu!\r\n");

```

```

435.         scanf_s("%i", &userOption);
436.
437.         switch (userOption)
438.         {
439.         case 1:
440.             length();//code to launch Length
441.             break;
442.         case 2:
443.             normalize();//code to launch Normalize
444.             break;
445.         case 3:
446.             distance();//code to make launch Distance between
two points
447.             break;
448.         case 4:
449.             crossProduct();//code to launch Cross product of
two vectors
450.             break;
451.         case 5:
452.             matrix();//code to launch Multiply a (4x4) and
(4x1) column matrix
453.             break;
454.         case 6:
455.             mean();//code to launch Mean
456.             break;
457.         case 7:
458.             median();//code to launch Median
459.             break;
460.         case 8:
461.             mode();//code to launch Mode
462.             break;
463.         case 9:
464.             standardDeviation();//code to launch Standard
Deviation
465.             break;
466.         case 10:
467.             unionIntersection();//code to launch Union and
Intersection of two sets
468.             break;
469.
470.         default:
471.             printf("Not a Valid Choice!\r\n");
472.             printf("Choose again!\r\n");
473.             break;
474.         }
475.
476.     }
477.     while (userOption != 8);
478.     return 0;
479. }

```

# Scott\_Howells\_1701604\_Assignment

---

## GRADEMARK REPORT

---

FINAL GRADE

GENERAL COMMENTS

**Instructor**

50/100

---

PAGE 1

---

PAGE 2

---

PAGE 3

---

PAGE 4

---

PAGE 5

---

PAGE 6



### Comment 1

Try this with a pair of 3D vectors i.e.  $(x_1, y_1, z_1)$  and  $(x_2, y_2, z_2)$



### Comment 2

Test values used could be better chosen.

---

PAGE 7



### Comment 3

Wrong calculation for cross product.

This is the dot product!!

---

PAGE 8



### Comment 4

This is confused. A 4x4 matrix will have 16 elements.

---

PAGE 9

---

PAGE 10

---

PAGE 11

---

**Comment 5**

This code is a little simplistic. It needs to cope with a general input of elements in sets of varying size.

The mode calculation also needs to work with sets with more than one mode or no mode.

GRADING FORM: SMU GENERIC\_GJ

**SCOTT HOWELLS****49.50****RESEARCH-INFORMED LITERATURE**

Refs could be better. Try to use established texts.  
Evidence of investigation into relevant topics and methods.

**2.5**

## KNOWLEDGE AND UNDERSTANDING



Good understanding of essential principles - some ideas need further work. some errors with code.  
Cross product is documented as a dot product!

**7**

## ANALYSIS



Evidence of analysis of problem and interpretation of problem into code. Tests included. Some ideas need further investigation (e.g. cross product, standard deviation, matrices etc.)

**7**

## PRACTICAL APPLICATION AND DEP.



Matrices code faulty, as is std dev, cross product not implemented correctly. Otherwise a fair attempt. Evidence of good SE practices here. Use comments more in your code.

**30.5**

## SKILLS FOR PROFESSIONAL PRAC.



Report is structured - evidence of test schedule - please document test data values here. Refs could be of a better quality and quantity.

**2.5**