Table

Description automatically generated

|  |  |
| --- | --- |
| Name : Muhammad Junaid Saleem Qadri | Enrolment No: 02-131202-057 |

Table

Description automatically generated

ALGORITHM :-

LIST is a integer array where its SIZE comes from user. Following algorithm inserts a number (int) from user into the Kth position in the LIST.

1.[Initialize counter.] Set K := 0.

2.Repeat Steps 3 and 4 while SIZE > K.

3. [Store number into LIST.] Set LIST[K] := ReadLine().

4. [Increment counter.] Set K := K + 1.

[End of Step 2 loop.]

5. Calculate mean by call the function and Store it in a double variable meanScore with 2 decimal digit.

Set meanScore := mean(LIST);

6. Print meanScore.

7. Calculate SD by call the function (deviation) and Store it in a double variable Sdeviation with 2 decimal digit.

Set Sdeviation := deviation(LIST);

8. Print Sdeviation.

9. Calculate the letter grade corresponding to each numeric score(LIST) by calling a function letterGrade and store it into char array (Grade).

Set char[] Grade := letterGrade (LIST, meanScore, Sdeviation) .

10. Print Grade.

11.Exit.

**Creating Function**

A) Create function of Mean with double return type:

double mean(int[] list)

1. Set double sum := 0
2. [Initialize counter.] Set i := 0.
3. Reapeat step 4 and 5 while list.Length > i.
4. [Calculate Sum ] Set sum := list[i] + sum.
5. [Increment counter] Set i := i + 1.

[End of Step 3 loop.]

1. Return sum / list.Lenght .

B) Create function of Standard Deviation with double return type and 2 arguments:

double deviation(int[] list , Double mean)

1. Set double sum := 0 .
2. [Initialize counter.] Set i := 0.
3. Reapeat step 4 and 5 while list.Length > i.
4. Set sum =: ((list[i] – mean) \* (list[i] - mean)) + sum .
5. [Increment counter] Set i := i + 1.

[End of Step 3 loop.]

1. [Calculate SD and return]

Return Math.sqrt(sum / list.Length).

C) Creating a function of Letter Grading with 3 arguments and return type of char array:

Char[] letterGrade (int[] list , Double mean , Double dev)

1. [Creating char array] : char[] letter = new char[list.Length] .
2. Set sub1 := mean - (dev / 2) .
3. Set add1 := mean + (dev / 2) .
4. Set sub3 := mean - (3 \* dev / 2).
5. Set add3 := mean + (3 \* dev / 2).
6. [Initialize counter.] Set i := 0.
7. Reapeat step 8 to 13 while list.Length > i.
8. If (list[i] < sub3) : Set letter[i] = 'F' .
9. Else if (sub3 <= list[i] && list[i] < sub1) : Set letter[i] = 'D' .
10. Else if (sub1 <= list[i] && list[i] < add1) : Set letter[i] = 'C' .
11. Else if (add1 <= list[i] && list[i] < add3) : Set letter[i] = 'B' .
12. Else if (sub3 < = list[i]) : Set letter[i] = 'A' .
13. [Increment counter] Set i := i + 1.

[End of Step 7 loop.]

1. Return letter.

Cooding :-

static double mean(int[] list)

{

double sum = 0;

for(int i=0; i<list.Length; i++)

{

sum = list[i] + sum;

}

return sum / list.Length;

}

static Double deviation(int[] list, Double mean)

{

double sum = 0;

for(int i = 0; i<list.Length; i++)

{

sum = ((list[i] - mean) \* (list[i] - mean)) + sum;

}

return Math.Sqrt(sum / list.Length);

}

static char[] letterGrade(int[] list , Double mean , Double dev)

{

char[] letter = new char[list.Length];

Double sub1 = mean - (dev / 2);

Double add1 = mean + (dev / 2);

double sub3 = mean - (3 \* dev / 2);

double add3 = mean + (3 \* dev / 2);

for (int i = 0; i<list.Length; i++)

{

if(list[i] < sub3)

{

letter[i] = 'F';

}

else if (sub3 <= list[i] && list[i] < sub1)

{

letter[i] = 'D';

}

else if (sub1<=list[i] && list[i] < add1)

{

letter[i] = 'C';

}

else if (add1 <= list[i] && list[i] < add3)

{

letter[i] = 'B';

}

else if (sub3 <= list[i])

{

letter[i] = 'A';

}

}

return letter;

}

static void Main(string[] args)

{

Console.Write("How much real number store in a set : ");

int SIZE = int.Parse(Console.ReadLine());

int[] list = new int[SIZE];

for(int K = 0; K<SIZE; K++)

{

Console.Write("\nEnter "+(K+1)+" Numeric score = ");

list[K] = int.Parse(Console.ReadLine());

}

Double meanScore = Convert.ToDouble(String.Format("{0:0.##}",mean(list)));

Console.WriteLine("\n\nMean Score (m) is = "+meanScore);

Double Sdeviation = Convert.ToDouble(String.Format("{0:0.##}",deviation(list, meanScore)));

Console.WriteLine("\nThe Standard Deviation is = "+ Sdeviation);

char[] Grade = letterGrade(list, meanScore, Sdeviation);

Console.WriteLine("\n----------------------------------------------");

Console.WriteLine("| S.No. | Numeric Score | Letter Grade |");

Console.WriteLine("----------------------------------------------");

for (int i = 0; i < list.Length; i++)

{

Console.WriteLine("|\t |\t\t |\t\t |");

Console.Write("| {0,-5}|", i + 1);

Console.Write(" \t {0,-10}|", list[i]);

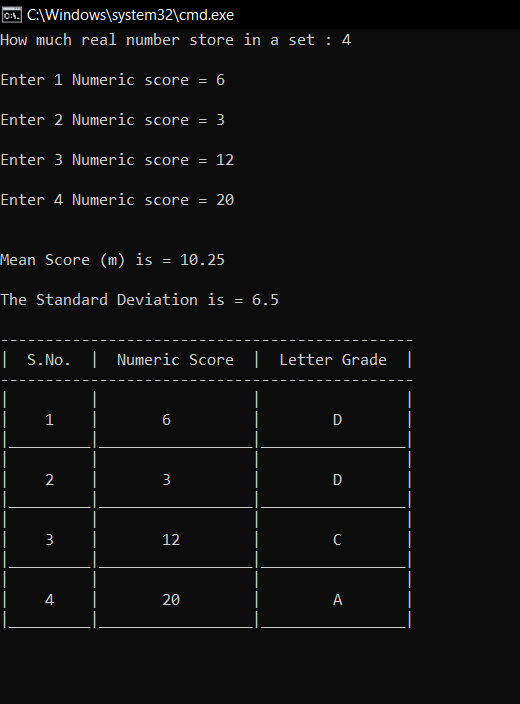
Console.WriteLine("\t {0,-8}|", Grade[i]);

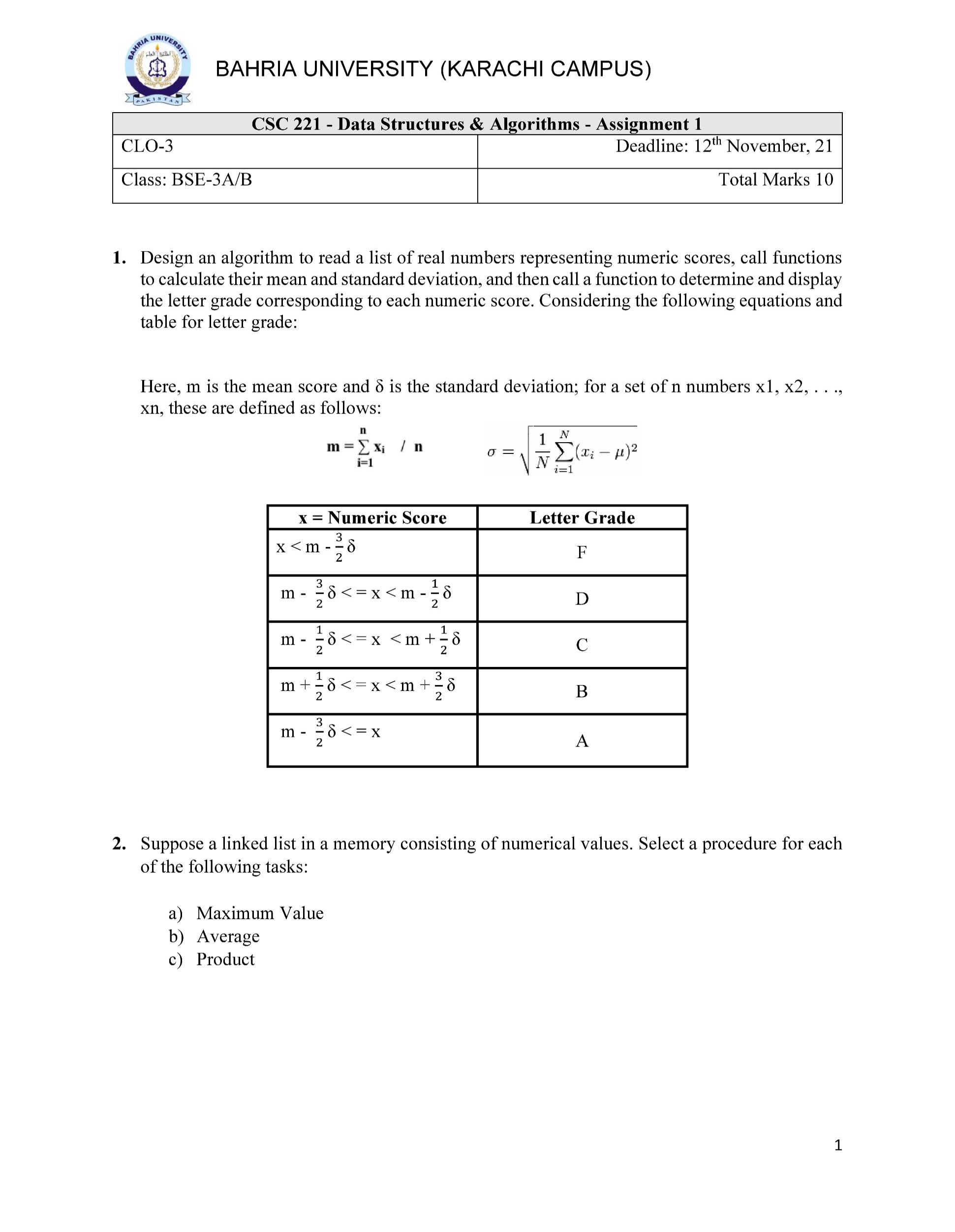
Console.WriteLine("|\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|");

}

}

Output :-





ALGORITHM :-  
a) Creating function of Maximum value with int return type:

int maxValue()

1. Set max := 0.
2. [Initialize counter.] Set Node := head .
3. Repeat Step 4 and 5 while Node == null.
4. if max < Node.data then Set max = Node.data .
5. [Increment counter] Set Node := Node.next .

[End of Step 3 loop.]

1. Return max.

b) Creating function of Average with double return type :

double average()

1. Set sum := 0 and count := 0.
2. [Initialize counter.] Set Node n := head .
3. Repeat Step 4 to 6 while n == null .
4. [Increment] count := count +1.
5. Set sum := n.data + sum .
6. [Increment counter] Set n := n.next .

[End of Step 3 loop.]

1. [Calculating Average] Set avg := sum / count.
2. Return avg .

c) Creating function of Product with int return type :

int Product()

1. Set product := 1.
2. [Initialize counter.] Set Node n := head .
3. Repeat Step 4 and 5 while n == null .
4. [Calculating Product] Set product := n.data \* product .
5. [Increment counter] Set n := n.next .

[End of Step 3 loop.]

1. Return product .

cooding :-

**Node Class**

public class Node

{

public int data;

public Node next;

public Node(int data)

{

this.data = data;

next = null;

}

}

**Link List Class**

public class Link\_List

{

public Node head;

public void printList()

{

for(Node n = head; n!=null; n = n.next)

{

Console.Write(n.data+" ");

}

}

public int maxValue()

{

int max = 0;

for(Node n =head; n!=null; n = n.next)

{

if (max < n.data)

{

max = n.data;

}

}

return max;

}

public double average()

{

double sum = 0;

int count = 0;

for(Node n=head; n!=null; n = n.next)

{

count++;

sum = n.data + sum;

}

double avg = sum / count;

return avg;

}

public int Product()

{

int product = 1;

for (Node n = head; n != null; n = n.next)

{

product = n.data \* product;

}

return product;

}

}

**Main Method**

static void Main(string[] args)

{

Link\_List list = new Link\_List();

list.head = new Node(5);

Node second = new Node(7);

Node third = new Node(9);

Node forth = new Node(3);

list.head.next = second;

second.next = third;

third.next = forth;

Console.Write("Linked List Array = ");

list.printList();

Console.WriteLine("\n\nmax Value is : " + list.maxValue());

Console.WriteLine("The average value is : "+list.average());

Console.WriteLine("The Product of all values is : " + list.Product());

Console.ReadLine();

}

output :-

Text

Description automatically generated