**Table of Contents**

|  |
| --- |
| 1. Introduction-------------------------------------------------------------------------------------------- 1 |
|  |
| 1. Programming Problems  * 3.1 Problem 1------------------------------------------------------------------------------------------1 - 5 * 3.2 Problem 2------------------------------------------------------------------------------------------ 5 -12 * 3.3 Problem 3-----------------------------------------------------------------------------------------12 - 17 |
| 1. Mixed Learning Programming  * 3.4 Online e-portfolio * Reflection-------------------------------------------------------------------------------------------------- 18 |
| 1. Conclusion------------------------------------------------------------------------------------------------19 |

**Ans to the Q. No- 1**

**Introduction:**

I am already familiar with pseudocode, flowcharts, and C programming. In programming, I am familiar with arrays, loops, and functions. I also learned how to draw a flowchart for nested if-else and loops. I also learn about mean, median, and standard deviation. Most importantly, I didn't know how to deal with kurtosis. I try my hardest to solve all of the problems.

**Ans to the Q. No- 2(3.1)**

**3.1 Problem 1:** Password Compliance: Here, password should be at least 8 character which have upper letter, lower letter, special digit and number.

**Pseudocode:**

define password

create a pass\_length variable and find password length

if password length is greater than 7

if password have upper case letter, then set up = 1

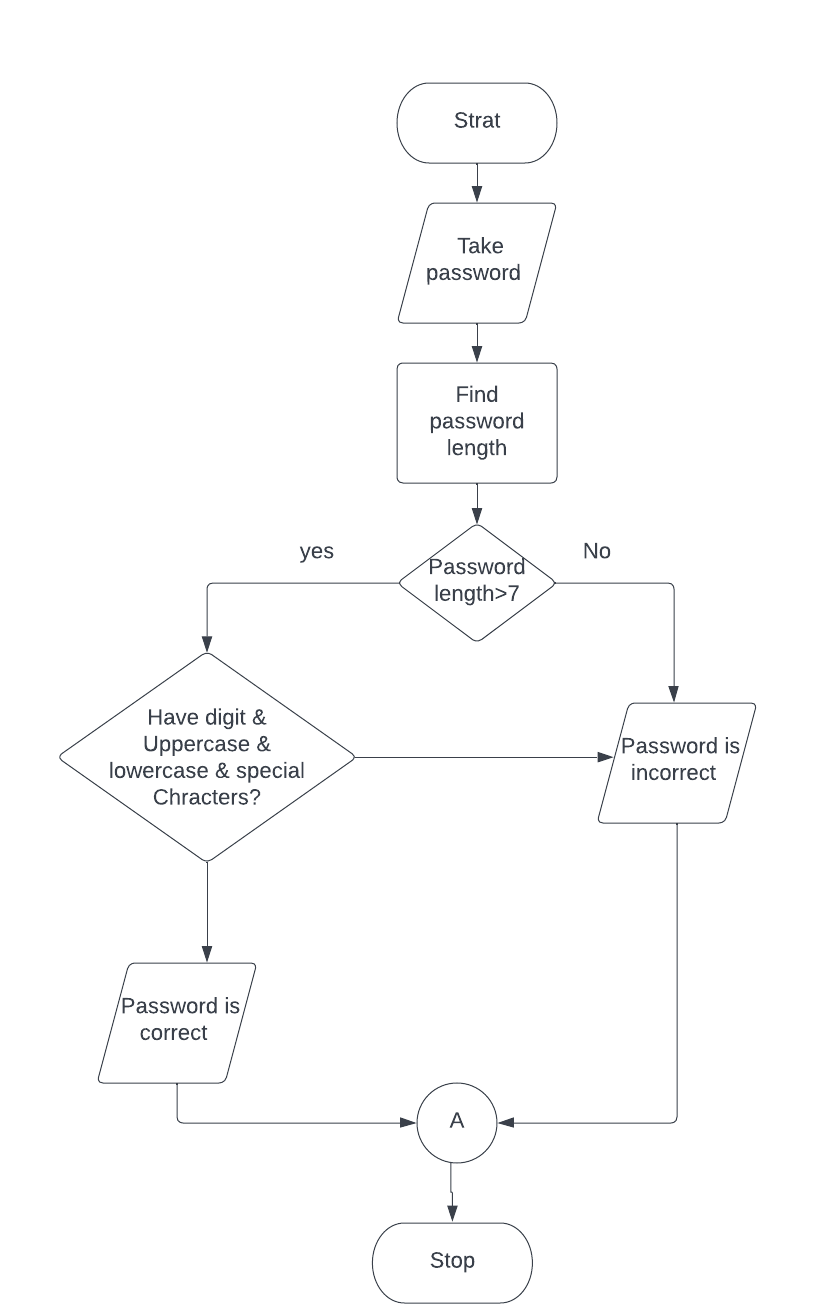
else if password have digit, then set digit =1

else if password has special character, then set dollar =1

return password id correct

else return password is incorrect

**Flowchart –**



**Code-**

**Snippet** – I will take a string input from user, then I will check the length of the string. If the length is greater than 7 and if the string has digit, upper- and lower-case letter, special character then it will print Password is correct otherwise print password is incorrect.

**Source Code-**

//declare header file

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

//The C <ctype. h> header file declares a set of functions to //classify (and transform) individual characters ex- check //isdigit(), isupper()

#include <ctype.h>

//define maximum size

#define MAXSIZE 100

//declare user define function

void pass\_read(char pass[]);

int pass\_check(char pass[]);

int

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

//taking pass array & integer type sum

char pass[MAXSIZE];

int sum;

//read input password using pass\_read() function with password //argument

pass\_read(pass);

//check password using pass\_check() function

sum = pass\_check(pass);

//if sum is true that's mean password follows all the condition //then it will print Password is correct

//else it will print password is incorrect

if (sum) {

printf("Your password is correct\n");

}

else {

printf("Your password is incorrect\n");

}

return 0;

}

//taking array input password from use

void

pass\_read(char pass[]) {

printf("Enter password:\n");

scanf("%s", pass);

}

int

pass\_check(char pass[]) {

int up = 0; //set up, digit, dollar is zero

int digit = 0;

int dollar = 0;

int i;

if(strlen(pass) > 7){ //find password's length and if length //is greater than 7

for (i = 0; i < strlen(pass); i++) {

//if there is uppercase then set up =1

if (isupper(pass[i])) {

up = 1;

}

//if there is digit then set digit =1

else if (isdigit(pass[i])) {

digit = 1;

}

//if there is a special character then set dollar =1

else if (pass[i] == '$') {

dollar = 1;

}

}

}//return up,digit,dolla value

return (up && digit && dollar);

}

**Code’s Screenshot:**



**Output –**

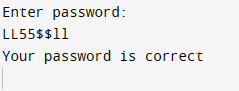


Figure: 1

**Another one**

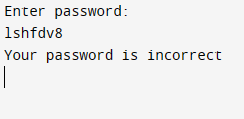


Figure: 1

**Ans to the Q. No- 2(3.2)**

**Problem 2:** Find largest & smallest word in a list & count vowel.

**Pseudocode:**

* Declare variables to store the largest and smallest words, and a variable to store the current word.
* Prompt the user to enter a list of words.
* Read in the first word and store it in the largest and smallest word variables.

- While the user has not reached the end of the input:

a. Read in the next word and store it in the current word variable.

b. If the current word is larger than the largest word, store it in the largest word variable.

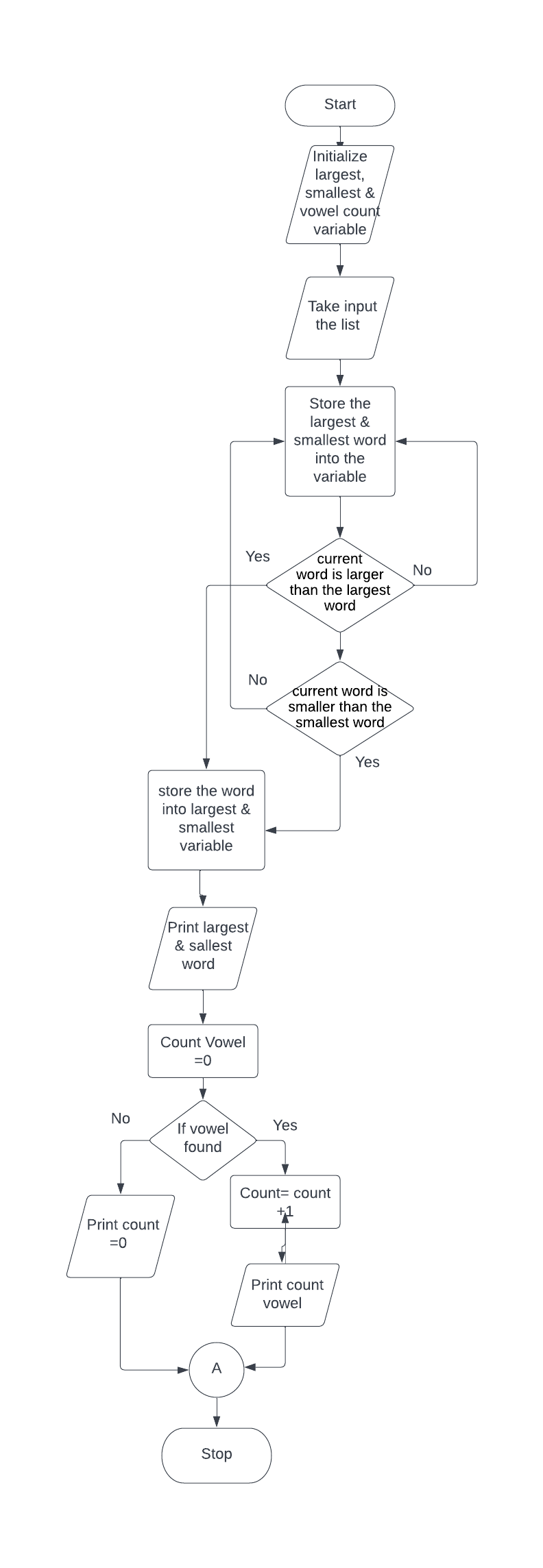
c. If the current word is smaller than the smallest word, store it in the smallest word variable.

- Print out the largest and smallest words.

- Find the largest and smallest word’s vowel. And count vowels if the word have “a,e,i,o,u” or ‘A,E,I,O,U’.

- Print the vowel count value.

**Flow Chart:**



**Code:**

**Snippets:**

First, I will take a longest and smallest variable and compare these to given country’s list for each element. Finding largest & smallest word & keep it into that array. Then find the vowel and count vowel for the largest & smallest word.

**Source Code-**

#include<stdio.h>

#include<string.h>

/\*GLOBAL METHOD DECLARATION\*/

int isVowel(char c);

int main()

{

/\*LINE 11 TO LINE 20 ARE USED TO DELCARE MY VARIABLES\*/

int i=0;

int j=0;

int k=0;

int a;

int minIndex=0;

int maxIndex=0;

int max=0;

int min=0;

char countries[100],lswords[100][100];

printf(" THE LARGEST AND THE SMALLEST WORD");

printf("\n PROGRAM ALSO OUTPUT THE NUMBER OF VOWELS IN BOTH WORDS\n");

printf("\n\tHERE'S A LIST OF 10 COUNTRIES:\n\t");

printf("\n\tAUSTRALIA, BRAZIL, CANADA, ARGENTINA, CHINA, FRANCE, GERMANY, Russia, INDONESIA, ITALY,\n\t");

printf("\n\tJAPAN, KOREA, MEXICO, INDIA, SAUDI ARABIA, SOUTH AFRICA, TURKEY, UK, USA\n\t");

printf("\nENTER 10 COUNTRIES FROM THE LIST \n");

printf("\n");

gets(countries); /\*THIS ALLOWS THE USER TO INPUT A STRING AND STORES THE DATA IN COUNTRIES\*/

while(countries[k]!='\0') /\*THIS IS USED TO SPLIT THE SENTENCES\*/

{

j=0;

while(countries[k]!=' '&&countries[k]!='\0')

{

lswords[i][j]=countries[k];

k++;

j++;

}

lswords[i][j]='\0';

i++;

if(countries[k]!='\0')

{

k++;

}

}

int len=i;

max=strlen(lswords[0]);

min=strlen(lswords[0]);

/\* USED AFTER SPLITTING THE LENGTH OF STRING\*/

for(i=0;i<len;i++)

{

a=strlen(lswords[i]);

if(a>max)

{

max=a;

maxIndex=i;

}

if(a<min)

{

min=a;

minIndex=i;

}

}

int counter1=0, counter2=0;

int v=0;

while(lswords[maxIndex][v]!='\0')

{

if(isVowel(lswords[minIndex][v]))

counter1++;

v++;

}

v=0;

while(lswords[minIndex][v]!='\0')

{

if(isVowel(lswords[minIndex][v]))

counter2++;

v++;

}

printf("\nTHE LARGEST WORD IS: %s\n", lswords[maxIndex]);

printf("\nTHE NUMBER OF VOWELS IN THE STRING '%s' is: %d\n", lswords[maxIndex],counter1);

printf("\nSMALLEST WORD IS: %s\n",lswords[minIndex]);

printf("\n NUMBER OF VOWELS IN THE STRING '%s' is: %d\n", lswords[minIndex],counter2);

return 0;

}

int isVowel(char c)

{

char vowels[]= {'a', 'e','i','o','u','A','E','I','O','U'};

int v;

for(v=0;v<=9;v++)

{

if(c==vowels[v])

return 1;

}

return 0;

}

**Code’s Screenshot:**



**Output**



Figure: 2

**Ans to the Q. No- 2(3.3)**

**Problem 3:** To find – Mean

- Median

- Kurtosis

- Standard deviation

Customer basket -> [22.3, 10.5, 60.5, 20.5, 22.5]

**Pseudocode:**

* Declare an array to hold the numbers and a variable to hold the array's size.
* Prompt the user to enter the numbers and store them in the array.
* Calculate the mean

summing all the numbers in the array and dividing by the size of the array.

* Sort the array in ascending order.
* Calculate the median by finding the middle value in the sorted array.

If the size of the array is odd, the median is the middle value.

If the size of the array is even, the median is the average of the two middle values.

* Calculate the standard deviation by first calculating the variance

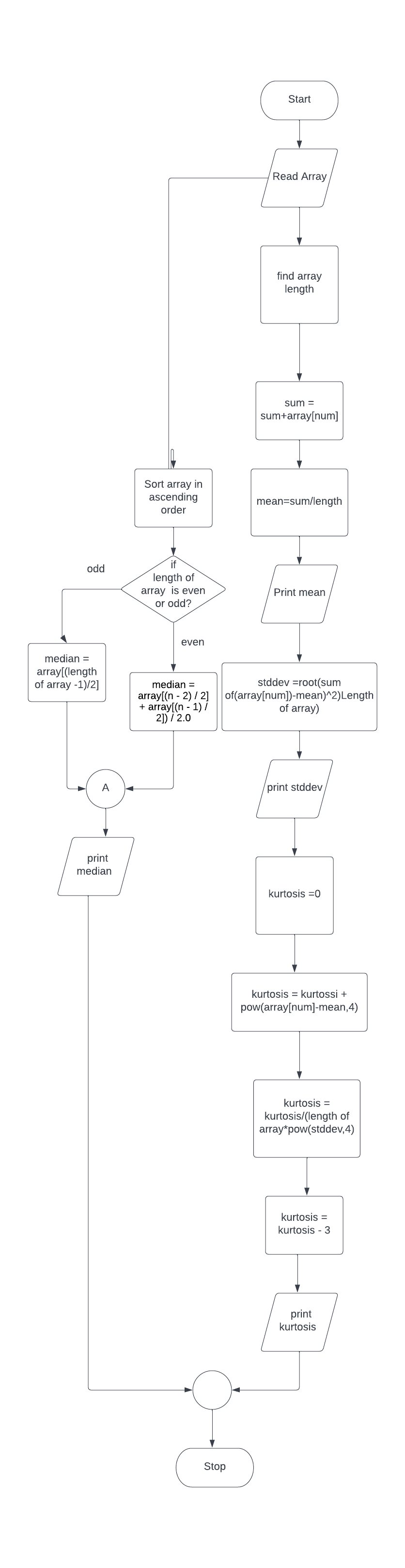
which is the sum of the squares of the differences between each number and the mean, divided by the size of the array.

Then, take the square root of the variance to get the standard deviation.

* Calculate the kurtosis by first calculation,

which is the sum of the fourth powers of the differences between each number and the mean, divided by the size of the array. Then, divide the fourth moment by the square of the variance to get the kurtosis.

**Flowchart-**



**Code:**

**Snippets:** At first, I will take an array that hold a floating number. Then I will calculate the mean by summing all the numbers in the array and dividing by the size of the array. Then I will be sorting the array in ascending order. To calculate median, I will take array size & if the size is odd then median will be middle value else median will be average of two values. To obtain standard deviation I will calculate the sum of the squares of the differences between each number and the mean, divided by the size of the array. Then, take the square root of the variance to get the standard deviation. Calculate Kurtosis I will take the sum of the fourth powers of the differences between each number and the mean, divided by the size of the array. Then, divide the fourth moment by the square of the variance to get the kurtosis.

**Source Code-**

#include <stdio.h>

#include <math.h>

// calculate the kurtosis of a data set

double kurtosis(double\* data, int n) {

// find the mean of the data

double mean = 0;

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

mean += data[i];

}

mean /= n;

// find the variance of the data

double var = 0;

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

var += (data[i] - mean) \* (data[i] - mean);

}

var /= n - 1;

printf("The Mean is = %lf\n", var);

// find the standard deviation of the data

double stdDevi = sqrt(var);

printf("Standard Deviation = %lf\n", stdDevi);

// find the kurtosis of the data

double kurtosis = 0;

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

kurtosis += pow(data[i] - mean, 4);

}

kurtosis /= n \* pow(stdDevi, 4);

kurtosis -= 3;

// Return the kurtosis

return kurtosis;

}

int compareDoubles(const void\* a, const void\* b) {

double x = \*(double\*)a;

double y = \*(double\*)b;

if (x < y) {

return -1;

} else if (x > y) {

return 1;

} else {

return 0;

}

}

// find the median of a data set

double median(double\* data, int n) {

// Sort the data

//sizeof to know the length

qsort(data, n, sizeof(double), compareDoubles);

// If the number of elements is odd, the median is the middle element

if (n % 2 == 1) {

return data[(n - 1) / 2];

}

//else the number of elements is even, the median is the average of the two middle elements

return (data[(n - 2) / 2] + data[(n - 1) / 2]) / 2.0;

}

int main(void) {

// Define a data set

double data[] = {22.3, 10.5, 60.5, 20.5, 22.5};

// Find the median of the data

double medianValue = median(data, 5);

// Print the median

printf("The median is: %f\n", medianValue);

// Calculate the kurtosis of the data

double kurtosisValue = kurtosis(data, 5);

// Print the kurtosis

printf("The kurtosis is: %f\n", kurtosisValue);

return 0;

}

**Code’s ScreenShot:**



**Output –**

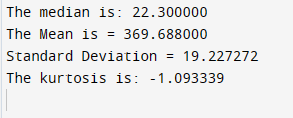


Figure: 3

**Ans to the Q. No- 3(3.5)**

* **The differences between the compiled and interpreted languages (Python vs C) –**

I completed a five-week mixed learning activity in which I learned both Python and C programming languages. After completing this activity, I gained a better understanding of the differences between compiled and interpreted languages, as well as a greater appreciation for Python and C.

The manner in which compiled and interpreted languages are executed is one of the most important distinctions. A compiler converts compiled languages, such as C, into machine code before they are executed (Wilson and Day, 2012). This means that the code is converted into a format that the computer's processor can directly execute, resulting in faster execution times. Interpreted languages, on the other hand, such as Python, are executed directly by an interpreter, which reads and executes the code line by line(Pilgrim and Willison, 2009). As a result, interpreted languages are slower than compiled languages, but they are also easier to write and debug.

* **Which language provides the most abstractions to make it easier to learn –**

In terms of abstraction and learning ease, I believe Python provides more abstractions than C. Python has a large standard library as well as a diverse set of built-in data types and functions that make it simple to perform common tasks without writing a lot of code. C, on the other hand, requires the programmer to be more explicit and to handle many low-level details, such as memory management and data type conversions (Ritchie, 1972), which can make it more difficult to learn for beginners (Stroustrup, 1983).

* **Using examples, illustrate which language provides the richest syntax, vocabulary, and subroutines etc. –**

However, when it comes to syntax, vocabulary, and subroutines, I believe C offers a more robust set of tools. C has a more expressive syntax than Python and a larger set of operators and control structures, allowing for more precise and efficient coding(Ritchie, 2009). C has a larger standard library as well as a diverse set of third-party libraries and frameworks that can be used to extend its capabilities (Kernighan and Ritchie, 1988). In terms of my learning experience, I found that Python was easier to learn and more enjoyable to work with than C. Python's clear and concise syntax and its rich standard library made it easier for me to understand and write code. However, I also enjoyed the challenge of learning C and appreciated the greater control and precision it afforded me.

* **Learning experience in both python and C and draw a conclusion** –

Overall, I would conclude that Python and C are both valuable programming languages that have their own strengths and weaknesses. Python is a great choice for beginners or for tasks that require rapid prototyping, while C is a more powerful and efficient language that is well suited for tasks that require more precise control and performance(IEEE, 1990).

**Ans to the Q. No- 4**

**Conclusion –**

I have mastered the logic of manipulating arrays and am also familiar with statistics mathematical aspects such as mean, median, kurtosis, and so on. However, I encountered numerous issues during password validation due to and the largest and smallest word in a list. I did, in fact, solve it and obtain the actual logic to implement them.

Reference

Wilson, D. Day, G.E., 2012. \*How To Be Awesome With Python.\* 1st ed. Manchester: Penguin.

Pilgrim, M. & Willison, S., 2009. Dive Into Python 3, Springer.

Kernighan, B., & Ritchie, D. (1988). The C Programming Language (2nd ed.). Prentice-Hall.

Stroustrup, B. (1983). The Evolution of C++. IEEE Transactions on Software Engineering, 9(7), 629-639.

Ritchie, D. (1972, June). The C Programming Language. Paper presented at the Conference on Data Systems Languages, San Jose, CA.

IEEE. (1990). C Programming Language. IEEE Std 1003.1-1990 (IEEE POSIX.1). IEEE Computer Society Press.

Ritchie, D. (2009, September 9). A Conversation with Dennis Ritchie. Video file.