**LAB 2 REPORT:**

**LAB OBJECTIVE:**

Show how varying complexity of the program changes runtime. Demonstrate this through finding the maximum subsequence sum using a cubic, quadratic, and linear function and show the times it took to complete them.

**REPORT STRUCTURE:**

Text is used to show analysis and high level programming.

**HOW THE CODE WORKS:**

**Define The Functions:**

There will be three functions, these three functions will each find the maximum subsequent sum for each vector fed into the functions. These functions will vary in complexity in the number of nested for-loops they have. In general each function works by starting at the beginning of the vector, summing the numbers ahead of it, and replacing its current max sum if the max sum found is larger than it. The complexity is triple nested (cubic), double nested (quadratic) and singly nested (linear). How each function works is explained in the comments included with the code:

A computer screen with text and numbers

Description automatically generated

Figure 1 – The cubic maxsum function.

A computer screen with green and white text

Description automatically generated

Figure 2 – The Quadratic MaxSum function

A screenshot of a computer

Description automatically generated

Figure 3 – The Linear MaxSum function

**Generate Vectors Full Of Random Numbers:**

A for loop is created to generate vectors filled with random numbers. The size of vectors is increased each time the loop is ran. Within the loop the results are reported each time.

A computer screen with many colorful text

Description automatically generated with medium confidence

Figure 4 – The Vector Generation

**Run Functions and Report the Results:**

The functions are run and a time stamp is generated before the function starts, and after the function ends. The difference between the two is used to determine the time it took the function to run. This time is reported along with the resulting maximum subsequent sum. Because the size of the vectors used increases within the loop, the time complexity of the various functions can be compared with the size of the vectors used.

A screen shot of a computer program

Description automatically generated

Figure 5 – The Result Reporting Section

**Results and Analysis:**

A black screen with white text

Description automatically generated

Figure 6 – A Sample Output Of The Program.

The program fulfills the objective. Interestingly each of the functions achieves the exact same result, but the times for each is dramatically different. While the cubic function ran in roughly 6-ten-thousandenth of a second, the quadratic function ran in 2-one-hundred-thousandeth of a second, and the linear function ran in 7 millionths of a second. The key takeaway is though one function may be more exact and take into account more scenarios, there is often a ‘good enough’ solution or a better more efficient solution that should be used instead. Such solutions produce approximate results while saving calculations, which may not be important on a small scale – but can become massively important as a program is scaled up.