ELEE-1147 Programming for Engineers

“The second Most Intelligent Species….”

July 15, 2022

Aakash Dutt Mathur

001093856

Aim: To extract information from the given audio files.

Introduction: We are provided with 2 audio files at different baud rates. These files contain the same message. The amount of data would be huge i.e. 122,880 samples per file. It would be very difficult for the audio engineer to deal with a single reading and manipulate it 20000 times instead dividing the data in blocks is a better option for handling such a huge amount of audio signals.

Task 1a:

Algorithm:

Step 1: The first step involves reading the signal. As there are binary files provided, the input data will be read from them instead of the STDIN. For this the binfileread function is made.
Step 2: The second step would be manipulating the data we got from reading that file. This step involves sorting the signal data, conversion to dits and dahs, ON/OFF, etc. This is done using the decimate function called in the main().
Step 3: Lastly using the binfilewrite, the discrete data can be rewritten and presented as an output to the viewer. Data would be returned in the form of an array.

Flow of information in the system:

Data In	**Process**	**Data Out**
Step 1: STDIN(Keyboard) (Not used)		**Step 1:** STDOUT(Terminal/VDU)
Step 2: Retrieve ground truth data from disk (Binary file in our case)		Step 2: Send verified data to disk
Step 3: Real data via communications channel		Step 3: Real-time data

Role of sub processes or functions called by main():

Process

Sub Process 1	Sub Process 2	Sub Process 3
binfileread.o	decimate.o	binfilewrite.o
binfileread() function contains: binfileread.cpp binfileread.hpp	decimate() function contains: decimate.cpp decimate.hpp	binfilewrite() function contains: binfilewrite.cpp binfilewrite.hpp

Both .cpp (source files) and .hpp (header) files are compiled to produce .o (object) files.
These object files are further linked to produce .exe or the executable file for the program to run.

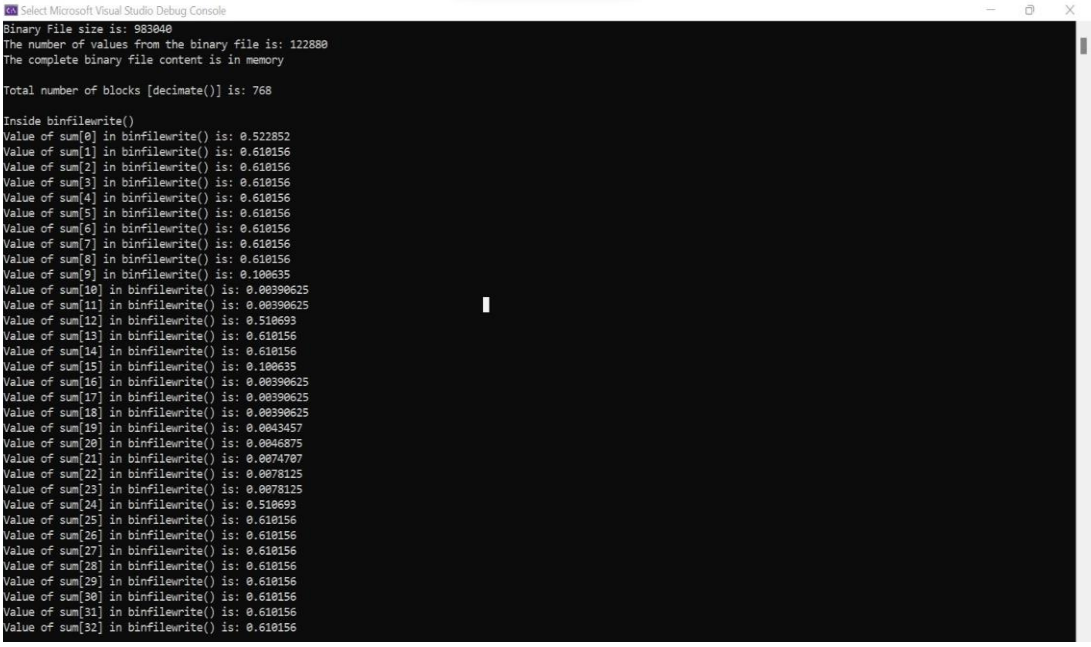
(.cpp & .hpp) -------------linker--------------→(.o) ------------Compiler---------→.exe (executable file)

Task 1b:

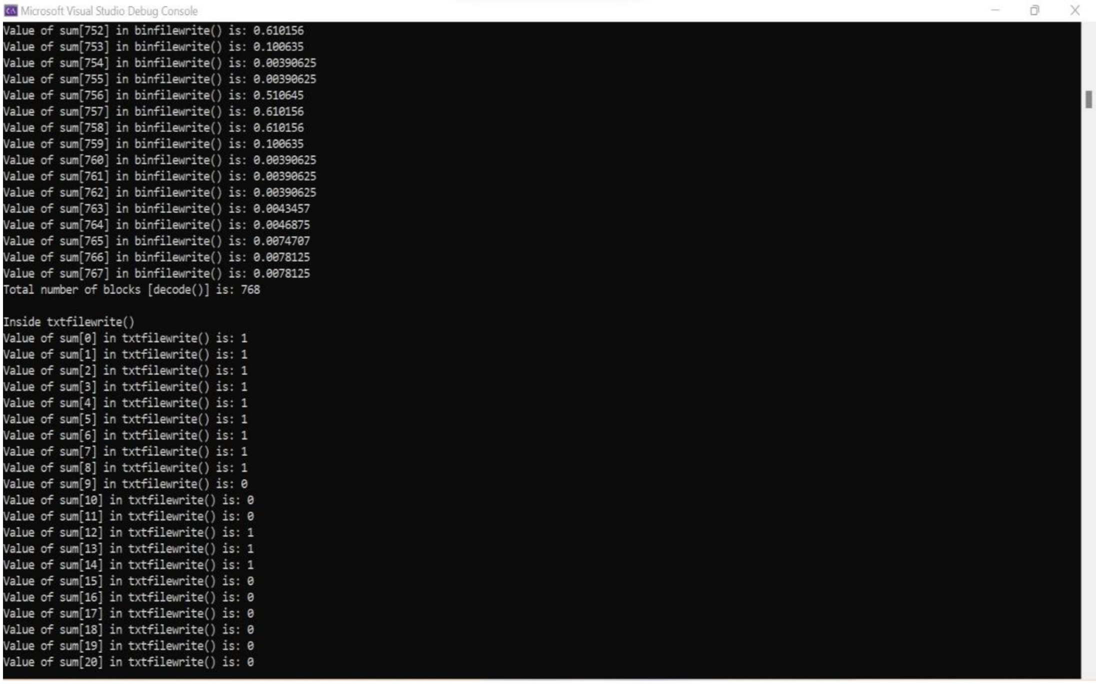
IDE Used: Visual studio 2022

The .cpp files or the source files are placed in the source folder in an empty project, and the header files are placed in the header folder.

The project is then built and run, and the output is:



```
Binary File size is: 983040
The number of values from the binary file is: 122880
The complete binary file content is in memory

Total number of blocks [decimate()] is: 768

Inside binfilewrite()
Value of sum[0] in binfilewrite() is: 0.522852
Value of sum[1] in binfilewrite() is: 0.610156
Value of sum[2] in binfilewrite() is: 0.610156
Value of sum[3] in binfilewrite() is: 0.610156
Value of sum[4] in binfilewrite() is: 0.610156
Value of sum[5] in binfilewrite() is: 0.610156
Value of sum[6] in binfilewrite() is: 0.610156
Value of sum[7] in binfilewrite() is: 0.610156
Value of sum[8] in binfilewrite() is: 0.610156
Value of sum[9] in binfilewrite() is: 0.100635
Value of sum[10] in binfilewrite() is: 0.00390625
Value of sum[11] in binfilewrite() is: 0.00390625
Value of sum[12] in binfilewrite() is: 0.510693
Value of sum[13] in binfilewrite() is: 0.610156
Value of sum[14] in binfilewrite() is: 0.610156
Value of sum[15] in binfilewrite() is: 0.100635
Value of sum[16] in binfilewrite() is: 0.00390625
Value of sum[17] in binfilewrite() is: 0.00390625
Value of sum[18] in binfilewrite() is: 0.00390625
Value of sum[19] in binfilewrite() is: 0.0043457
Value of sum[20] in binfilewrite() is: 0.0046875
Value of sum[21] in binfilewrite() is: 0.0074707
Value of sum[22] in binfilewrite() is: 0.0078125
Value of sum[23] in binfilewrite() is: 0.0078125
Value of sum[24] in binfilewrite() is: 0.510693
Value of sum[25] in binfilewrite() is: 0.610156
Value of sum[26] in binfilewrite() is: 0.610156
Value of sum[27] in binfilewrite() is: 0.610156
Value of sum[28] in binfilewrite() is: 0.610156
Value of sum[29] in binfilewrite() is: 0.610156
Value of sum[30] in binfilewrite() is: 0.610156
Value of sum[31] in binfilewrite() is: 0.610156
Value of sum[32] in binfilewrite() is: 0.610156
```



```
Value of sum[752] in binfilewrite() is: 0.610156
Value of sum[753] in binfilewrite() is: 0.100635
Value of sum[754] in binfilewrite() is: 0.00390625
Value of sum[755] in binfilewrite() is: 0.00390625
Value of sum[756] in binfilewrite() is: 0.510645
Value of sum[757] in binfilewrite() is: 0.610156
Value of sum[758] in binfilewrite() is: 0.610156
Value of sum[759] in binfilewrite() is: 0.100635
Value of sum[760] in binfilewrite() is: 0.00390625
Value of sum[761] in binfilewrite() is: 0.00390625
Value of sum[762] in binfilewrite() is: 0.00390625
Value of sum[763] in binfilewrite() is: 0.0043457
Value of sum[764] in binfilewrite() is: 0.0046875
Value of sum[765] in binfilewrite() is: 0.0074707
Value of sum[766] in binfilewrite() is: 0.0078125
Value of sum[767] in binfilewrite() is: 0.0078125
Total number of blocks [decode()] is: 768

Inside txtfilewrite()
Value of sum[0] in txtfilewrite() is: 1
Value of sum[1] in txtfilewrite() is: 1
Value of sum[2] in txtfilewrite() is: 1
Value of sum[3] in txtfilewrite() is: 1
Value of sum[4] in txtfilewrite() is: 1
Value of sum[5] in txtfilewrite() is: 1
Value of sum[6] in txtfilewrite() is: 1
Value of sum[7] in txtfilewrite() is: 1
Value of sum[8] in txtfilewrite() is: 1
Value of sum[9] in txtfilewrite() is: 0
Value of sum[10] in txtfilewrite() is: 0
Value of sum[11] in txtfilewrite() is: 0
Value of sum[12] in txtfilewrite() is: 1
Value of sum[13] in txtfilewrite() is: 1
Value of sum[14] in txtfilewrite() is: 1
Value of sum[15] in txtfilewrite() is: 0
Value of sum[16] in txtfilewrite() is: 0
Value of sum[17] in txtfilewrite() is: 0
Value of sum[18] in txtfilewrite() is: 0
Value of sum[19] in txtfilewrite() is: 0
Value of sum[20] in txtfilewrite() is: 0
```

Observation: Here we notice that the binfilewrite gives the true value of the output signal sample but as we move to the textfilewrite we obtain the exact binary values in the form of 0s and 1s which is perfect for our use.

Task 2:

The focus of this task is to make the data more manageable by maintaining the integrity and not losing any information. For this purpose, some diagrams can be used to analyse the data in the files.

For this purpose, we use MATLAB to plot the various graphs for both the audio files and try to spot the dits and dahs.

Visualising the data:

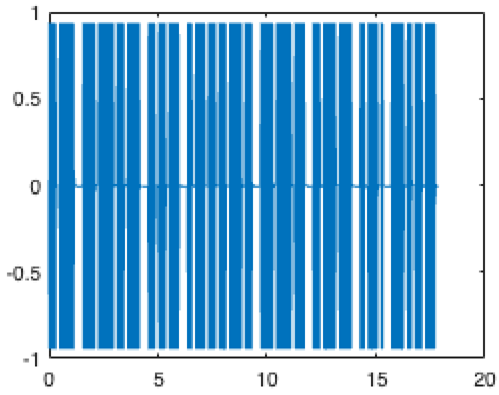
Here the plotting of the binary files is done using MATLAB. This is done by keeping the frequency of the message (fs) 8kHz, opening the file and plotting the message inside it in the timeframe of 2.5 seconds.

For dolphins_20wpm.bin :

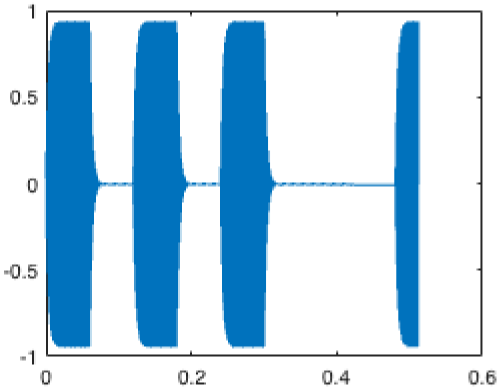
MATLAB Code and Output:

```
fid = fopen('dolphins_20wpm.bin');
data = fread(fid, 'double');
fclose(fid);
figure(1), plot(data)
```

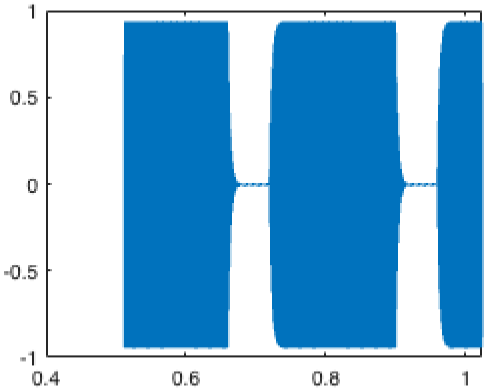
```
fs = 8000;
t = (0:(1/fs):(length(data)/fs)-(1/fs));
t = t';
figure(1), plot(data)
plot(t, data)
```



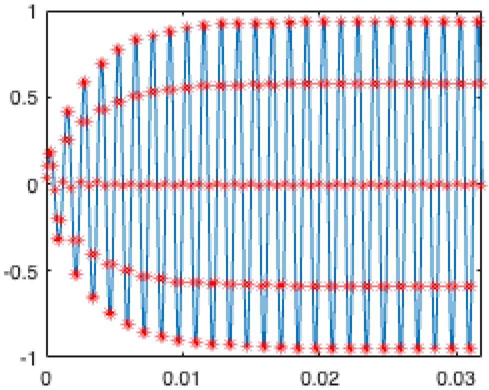
```
plot(t(1:4096),data(1:4096))
```



```
figure(2), plot(t(4097:8192), data(4097:8192));
```

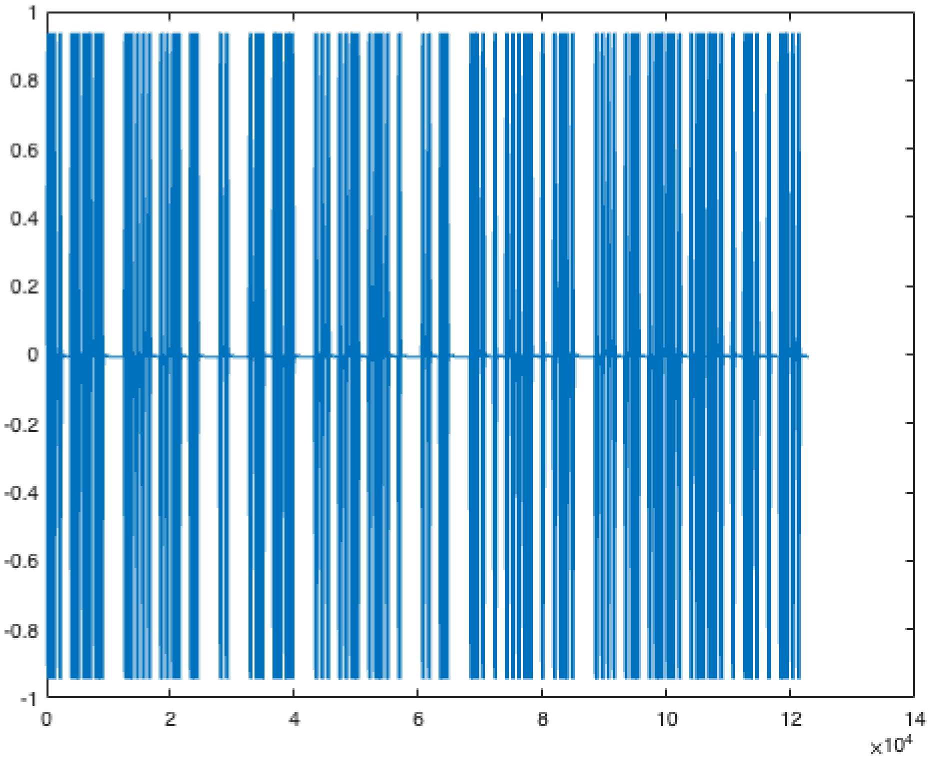


```
figure(3), plot(t(1:256), data(1:256)), hold on, plot(t(1:256), data(1:256), 'r*')
```

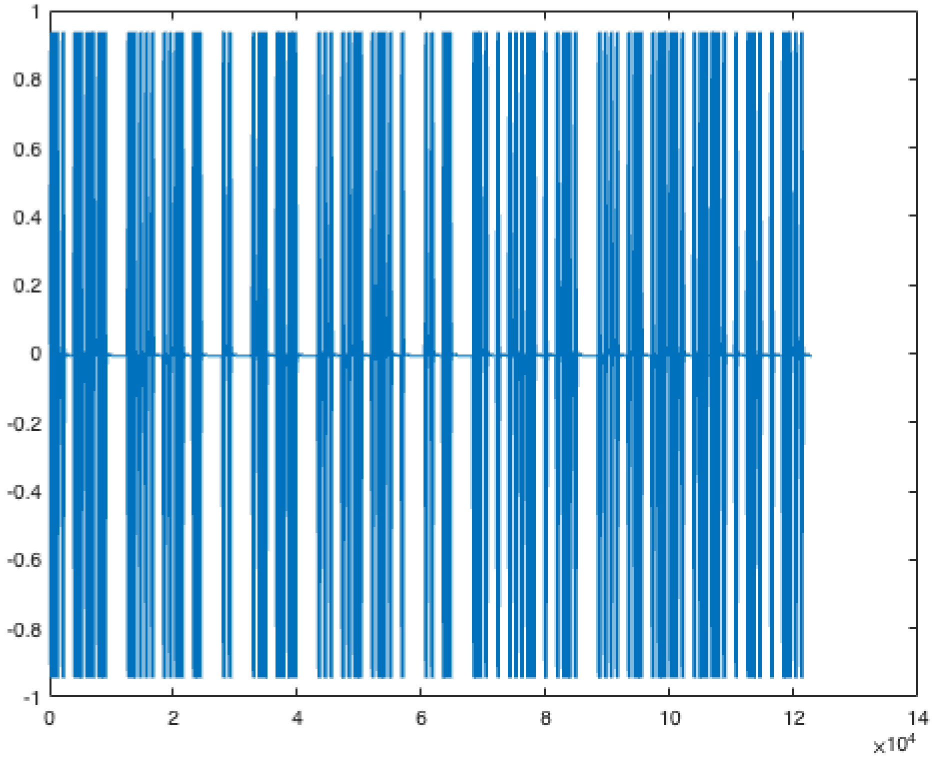


For boris_Nov13_20wpm.bin :

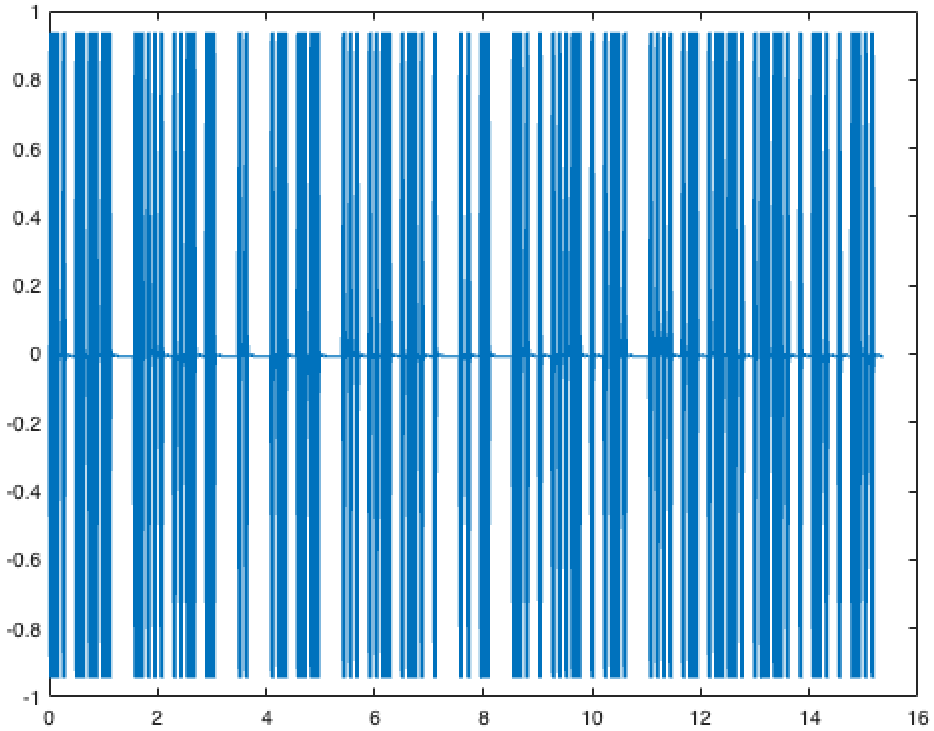
```
fid = fopen('boris_Nov13_20wpm.bin');
data = fread(fid, 'double');
fclose(fid);
figure(1), plot(data)
```



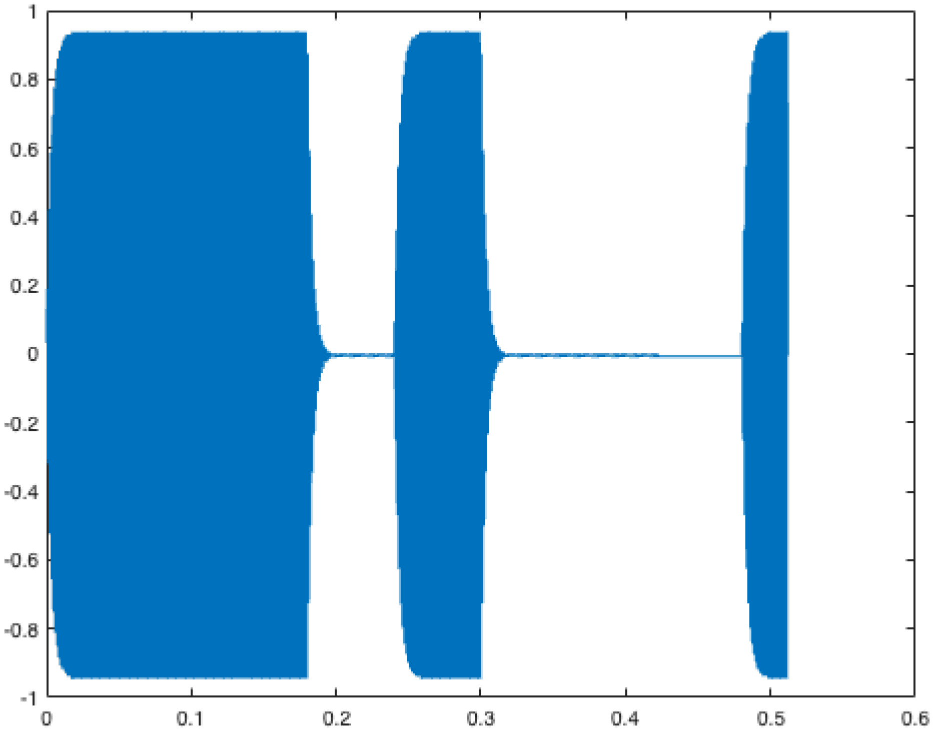
```
fs = 8000;
t = (0:(1/fs):(length(data)/fs)-(1/fs));
t = t';
figure(1), plot(data)
```


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
2
4
6
8
10
12
14
×10^4

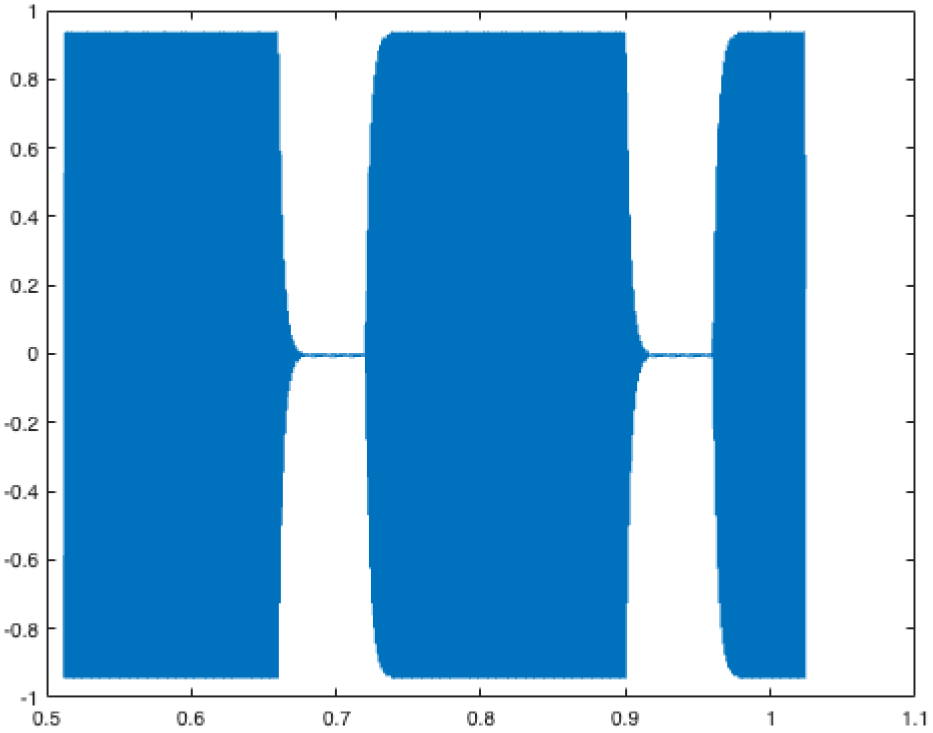
```
plot(t, data)
```


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
2
4
6
8
10
12
14
16

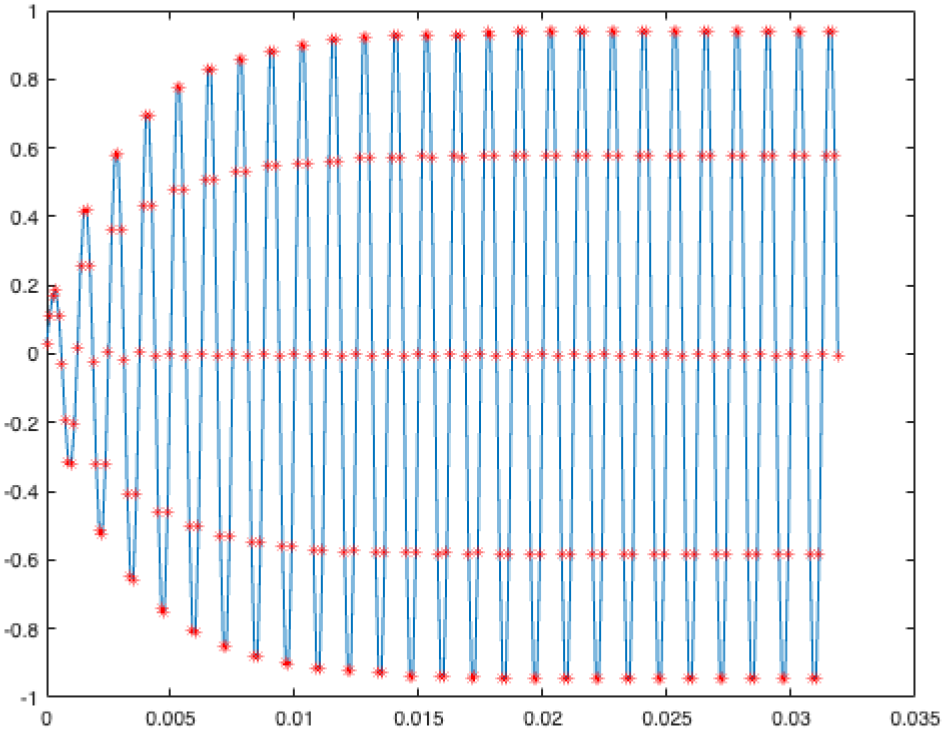
```
plot(t(1:4096),data(1:4096))
```



```
figure(2), plot(t(4097:8192), data(4097:8192));
```

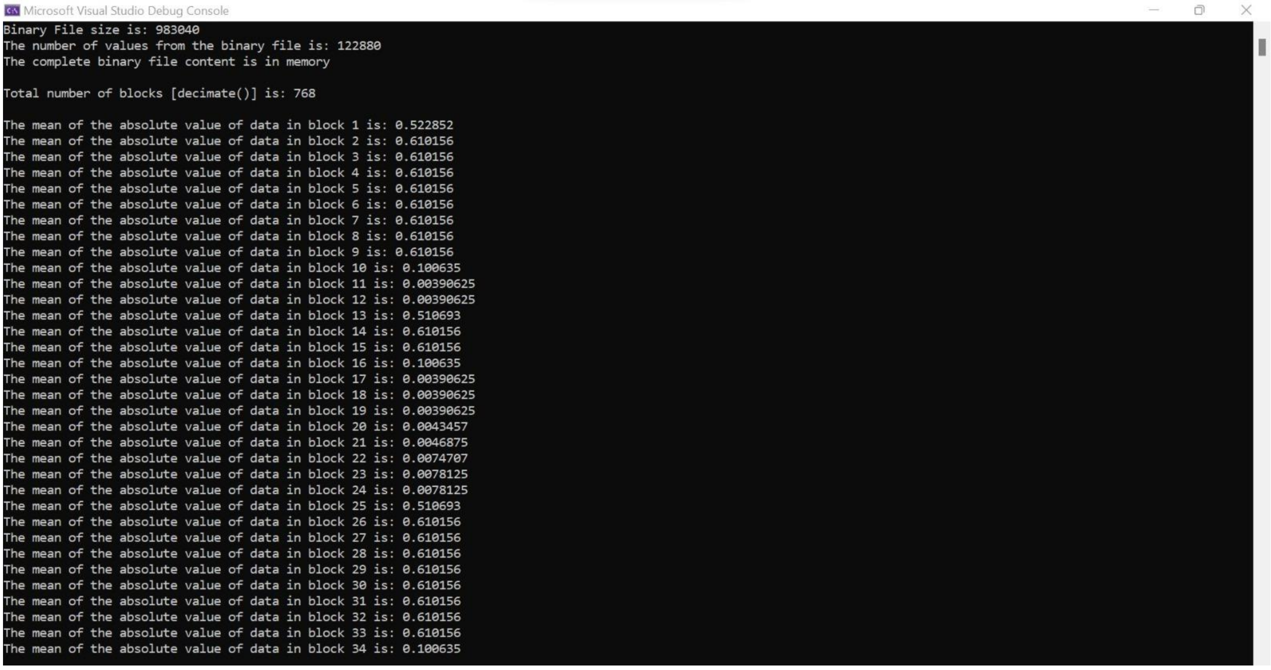


```
figure(3), plot(t(1:256), data(1:256)), hold on, plot(t(1:256), data(1:256), 'r*')
```



Manipulation & Sorting:

Here, the data manipulation is done according to the value of each sample as shown in the output.



```
Binary File size is: 983040
The number of values from the binary file is: 122880
The complete binary file content is in memory

Total number of blocks [decimate()] is: 768

The mean of the absolute value of data in block 1 is: 0.522852
The mean of the absolute value of data in block 2 is: 0.610156
The mean of the absolute value of data in block 3 is: 0.610156
The mean of the absolute value of data in block 4 is: 0.610156
The mean of the absolute value of data in block 5 is: 0.610156
The mean of the absolute value of data in block 6 is: 0.610156
The mean of the absolute value of data in block 7 is: 0.610156
The mean of the absolute value of data in block 8 is: 0.610156
The mean of the absolute value of data in block 9 is: 0.610156
The mean of the absolute value of data in block 10 is: 0.100635
The mean of the absolute value of data in block 11 is: 0.00390625
The mean of the absolute value of data in block 12 is: 0.00390625
The mean of the absolute value of data in block 13 is: 0.510693
The mean of the absolute value of data in block 14 is: 0.610156
The mean of the absolute value of data in block 15 is: 0.610156
The mean of the absolute value of data in block 16 is: 0.100635
The mean of the absolute value of data in block 17 is: 0.00390625
The mean of the absolute value of data in block 18 is: 0.00390625
The mean of the absolute value of data in block 19 is: 0.00390625
The mean of the absolute value of data in block 20 is: 0.0043457
The mean of the absolute value of data in block 21 is: 0.0046875
The mean of the absolute value of data in block 22 is: 0.0074707
The mean of the absolute value of data in block 23 is: 0.0078125
The mean of the absolute value of data in block 24 is: 0.0078125
The mean of the absolute value of data in block 25 is: 0.510693
The mean of the absolute value of data in block 26 is: 0.610156
The mean of the absolute value of data in block 27 is: 0.610156
The mean of the absolute value of data in block 28 is: 0.610156
The mean of the absolute value of data in block 29 is: 0.610156
The mean of the absolute value of data in block 30 is: 0.610156
The mean of the absolute value of data in block 31 is: 0.610156
The mean of the absolute value of data in block 32 is: 0.610156
The mean of the absolute value of data in block 33 is: 0.610156
The mean of the absolute value of data in block 34 is: 0.100635
```

As it is very difficult to reach out to every sample, this task of sorting is done with the help of if/else statement.

A standard threshold value is set. Each sample will be trimmed, sorted or judged according to this value. We set this value to be “0.5”.

Case A: if the value of the audio sample is less than 0.5, the signal will be considered as “OFF”

Case B: Else if the value of the sample is equal to or greater than 0.5 then the signal will be considered as “ON”

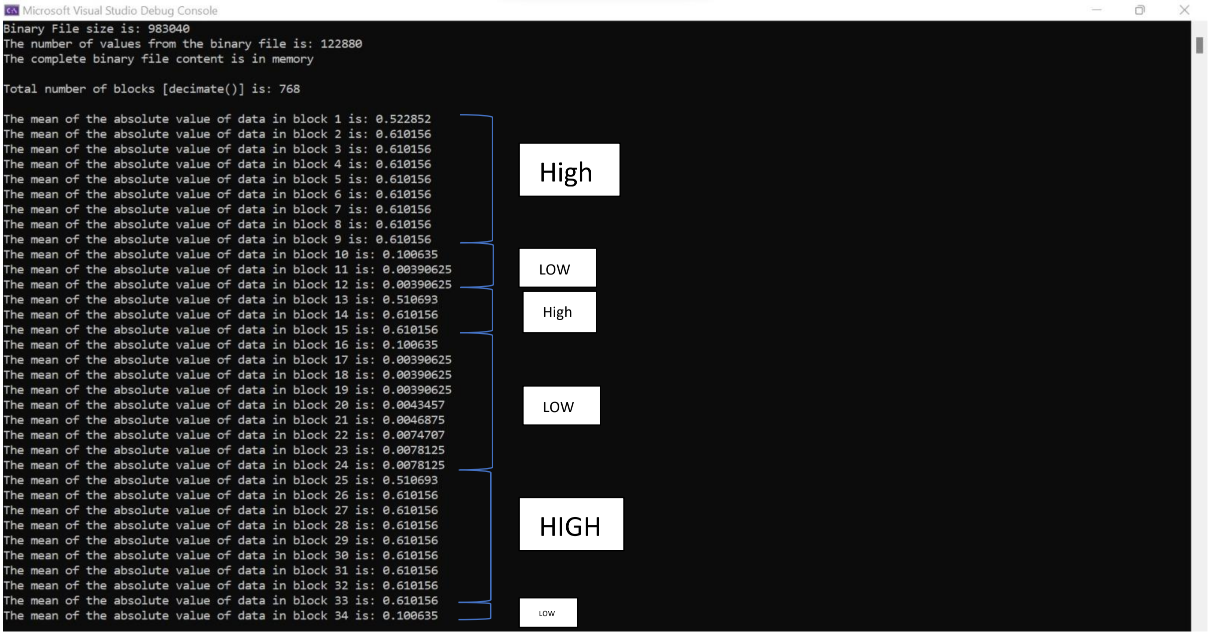
Code for if/else in the getContentFromSignal.cpp:

Void getContentFromSignal::checkONandOFF(vector<double>m, int j){

```
        if (mean[j] > 0.5)

        {

                countON++;

                countOFF = 0;

        }else{

                countOFF++;

                countON = 0;

                }

        }
```

Conclusion from task 2: This will help us to manually sort the data into signal ON or signal OFF.

Manual Interpretation of HIGHs and LOWs:



```
Binary File size is: 983040
The number of values from the binary file is: 122880
The complete binary file content is in memory

Total number of blocks [decimate()] is: 768

The mean of the absolute value of data in block 1 is: 0.522852
The mean of the absolute value of data in block 2 is: 0.610156
The mean of the absolute value of data in block 3 is: 0.610156
The mean of the absolute value of data in block 4 is: 0.610156
The mean of the absolute value of data in block 5 is: 0.610156
The mean of the absolute value of data in block 6 is: 0.610156
The mean of the absolute value of data in block 7 is: 0.610156
The mean of the absolute value of data in block 8 is: 0.610156
The mean of the absolute value of data in block 9 is: 0.610156
The mean of the absolute value of data in block 10 is: 0.100635
The mean of the absolute value of data in block 11 is: 0.00390625
The mean of the absolute value of data in block 12 is: 0.00390625
The mean of the absolute value of data in block 13 is: 0.510693
The mean of the absolute value of data in block 14 is: 0.610156
The mean of the absolute value of data in block 15 is: 0.610156
The mean of the absolute value of data in block 16 is: 0.100635
The mean of the absolute value of data in block 17 is: 0.00390625
The mean of the absolute value of data in block 18 is: 0.00390625
The mean of the absolute value of data in block 19 is: 0.00390625
The mean of the absolute value of data in block 20 is: 0.0043457
The mean of the absolute value of data in block 21 is: 0.0046875
The mean of the absolute value of data in block 22 is: 0.0074707
The mean of the absolute value of data in block 23 is: 0.0078125
The mean of the absolute value of data in block 24 is: 0.0078125
The mean of the absolute value of data in block 25 is: 0.510693
The mean of the absolute value of data in block 26 is: 0.610156
The mean of the absolute value of data in block 27 is: 0.610156
The mean of the absolute value of data in block 28 is: 0.610156
The mean of the absolute value of data in block 29 is: 0.610156
The mean of the absolute value of data in block 30 is: 0.610156
The mean of the absolute value of data in block 31 is: 0.610156
The mean of the absolute value of data in block 32 is: 0.610156
The mean of the absolute value of data in block 33 is: 0.610156
The mean of the absolute value of data in block 34 is: 0.100635
```

Task 3:

Now, we must implement the code or the function in task 2.

For this, decimate().cpp must be altered in order to fulfil our needs and manipulate the code according to “Morse norms”.

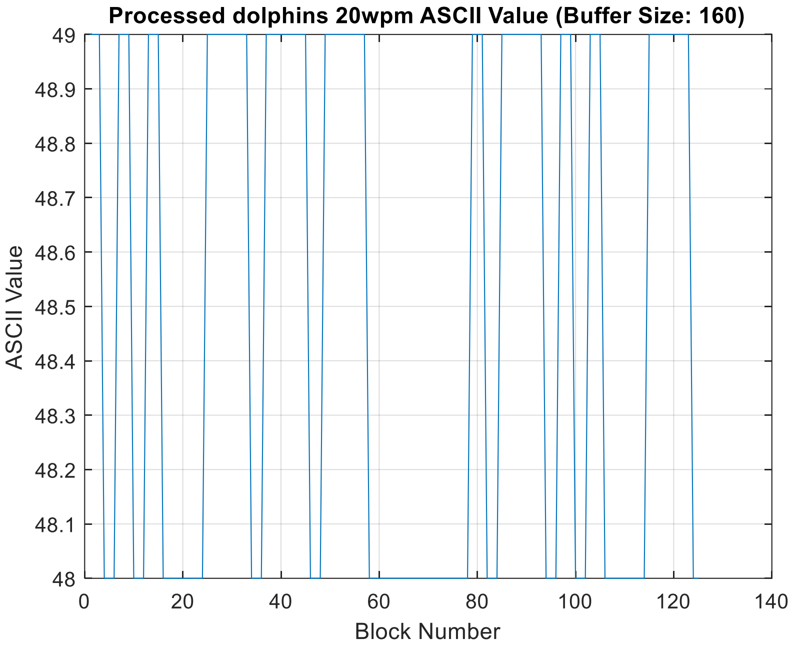
Additional Code for decimate.cpp:

```
//Block of code for calculating the dot and dash values using the baud rate

double calBaudRate = size / BUFFER_SIZE;

   int dotValue = round((calBaudRate / 3) + 1);

   int dashValue = dotValue * 3;

   int IWS = dotValue * 7;


// Lines of code used to check whether the signal is ON or OFF an then take action according to that.

getContent.checkONandOFF(mean, j);

      getContent.takeAction(dashValue, dotValue, IWS);

GetContentFromSignal getContent;
```

```
getContent.displayContent();
```

Morse to English Conversion:

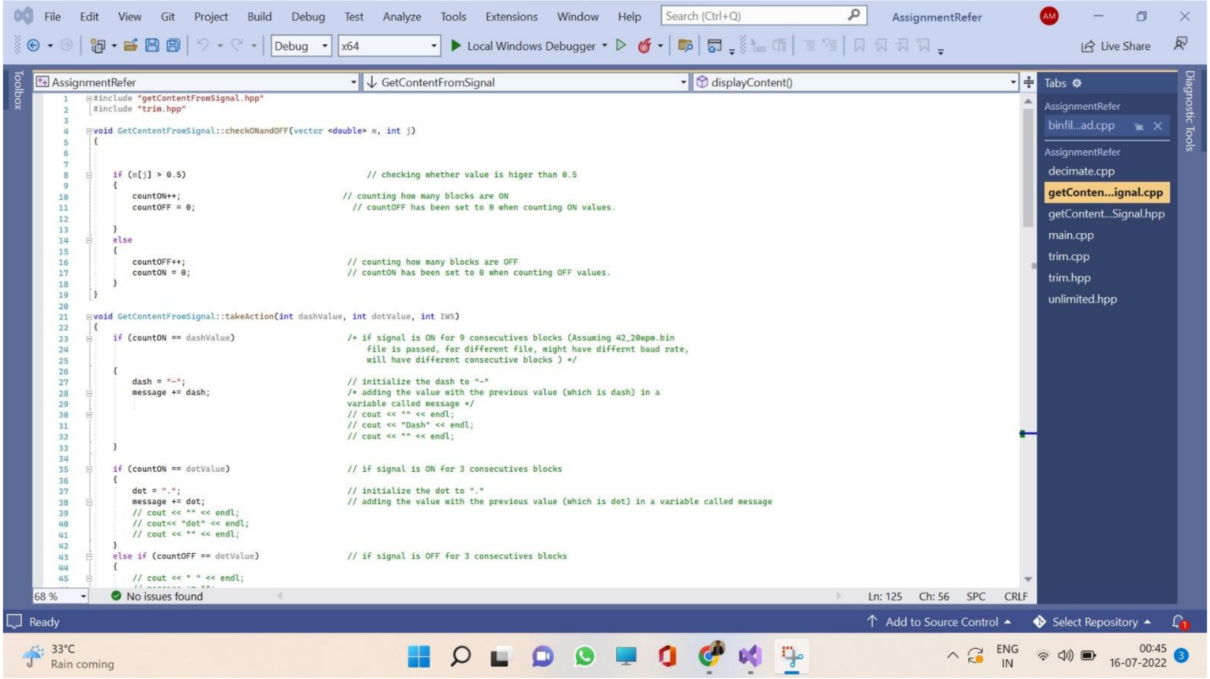
Till now the data has been gathered and categorised it into ONs and OFFs. Now according to the norms of morse code, dits and dahs are formed from the combination of ONs and OFFs.



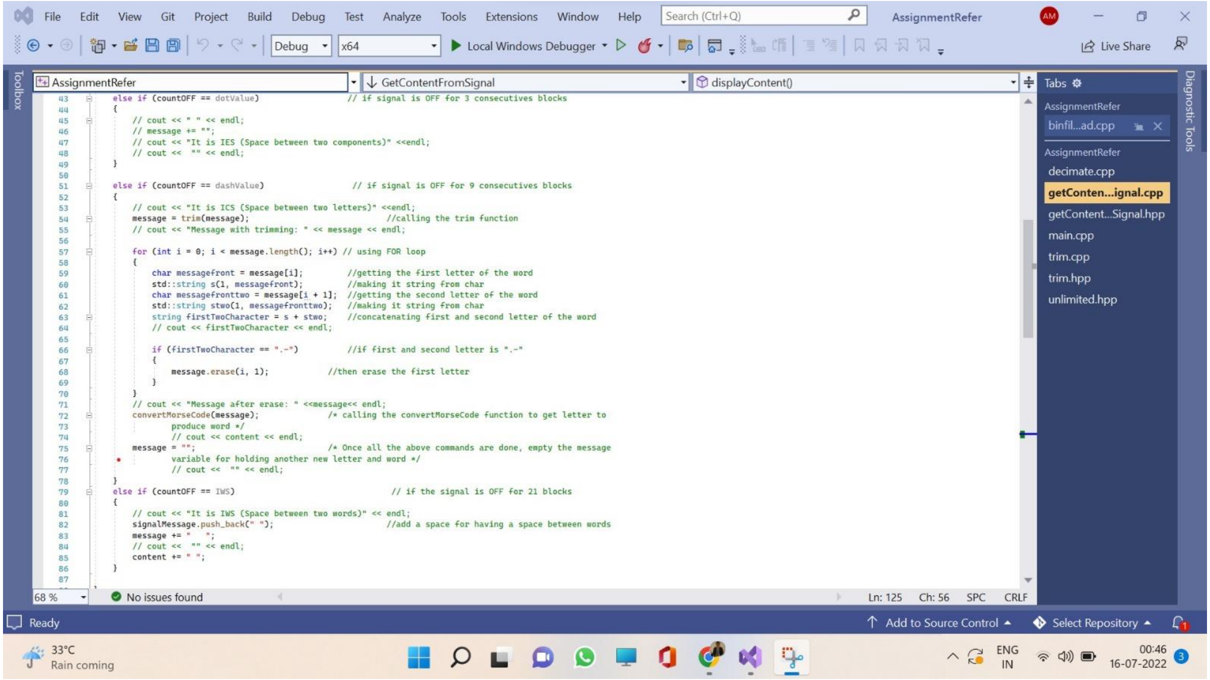
Like in Figure above the buffer size is taken 160 and we see that there are 3 dits in the starting. These are formed by the combination of HIGHs and LOWs.

For this conversion, another file getContentFromSignal.cpp along with getContentFromSignal.hpp is created, this file is included in the main project:

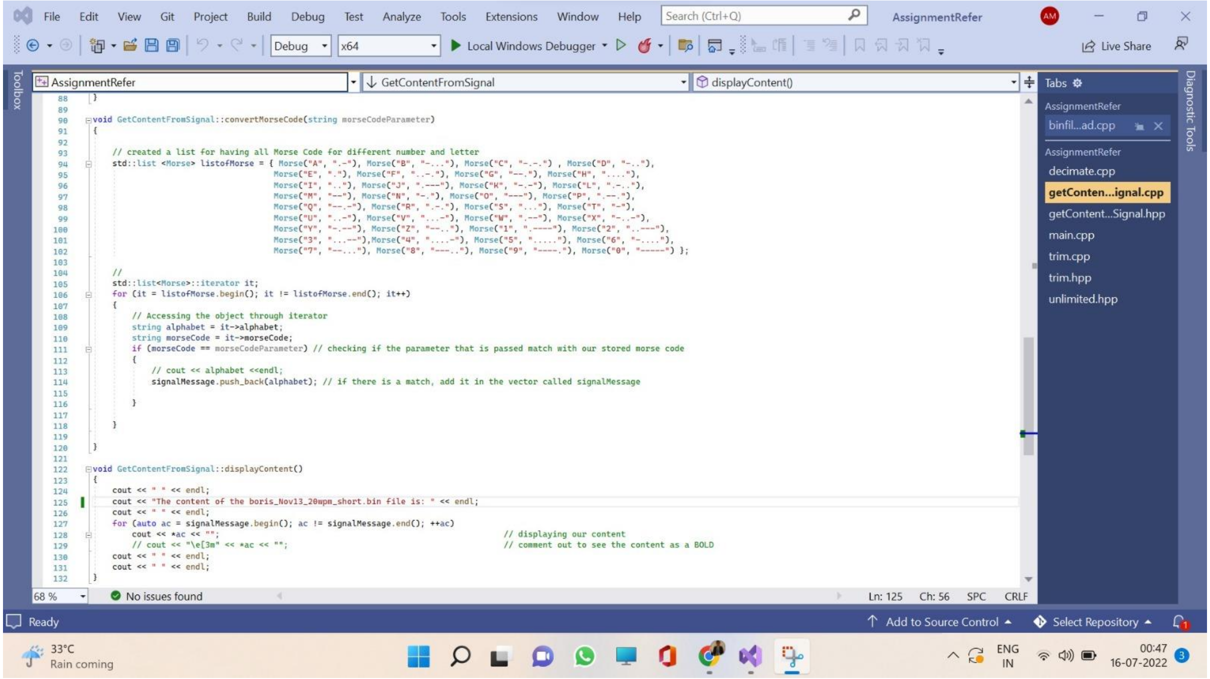
getContentFromSignal.cpp file:



```cpp
#include "getContentFromSignal.hpp"
#include "trim.hpp"

void GetContentFromSignal::checkONandOFF(vector <double> m, int j)
{


    if (m[j] > 0.5)                                     // checking whether value is higer than 0.5
    {
        countON++;                                  // counting how many blocks are ON
        countOFF = 0;                                 // countOFF has been set to 0 when counting ON values.

    }
    else
    {
        countOFF++;                                  // counting how many blocks are OFF
        countON = 0;                                 // countON has been set to 0 when counting OFF values.
    }
}

void GetContentFromSignal::takeAction(int dashValue, int dotValue, int IWS)
{
    if (countON == dashValue)                        /* if signal is ON for 9 consecutives blocks (Assuming 42_20wpm.bin
                                                         file is passed, for different file, might have differnt baud rate,
                                                         will have different consecutive blocks ) */

    {
        dash = "-";                                  // initialize the dash to "-"
        message += dash;                             /* adding the value with the previous value (which is dash) in a
                                                     variable called message */
                                                     // cout << "" << endl;
                                                     // cout << "Dash" << endl;
                                                     // cout << "" << endl;
    }

    if (countON == dotValue)                         // if signal is ON for 3 consecutives blocks
    {
        dot = ".";                                   // initialize the dot to "."
        message += dot;                              // adding the value with the previous value (which is dot) in a variable called message
        // cout << "" << endl;
        // cout<< "dot" << endl;
        // cout << "" << endl;
    }
    else if (countOFF == dotValue)                   // if signal is OFF for 3 consecutives blocks
    {
        // cout << " " << endl;
```

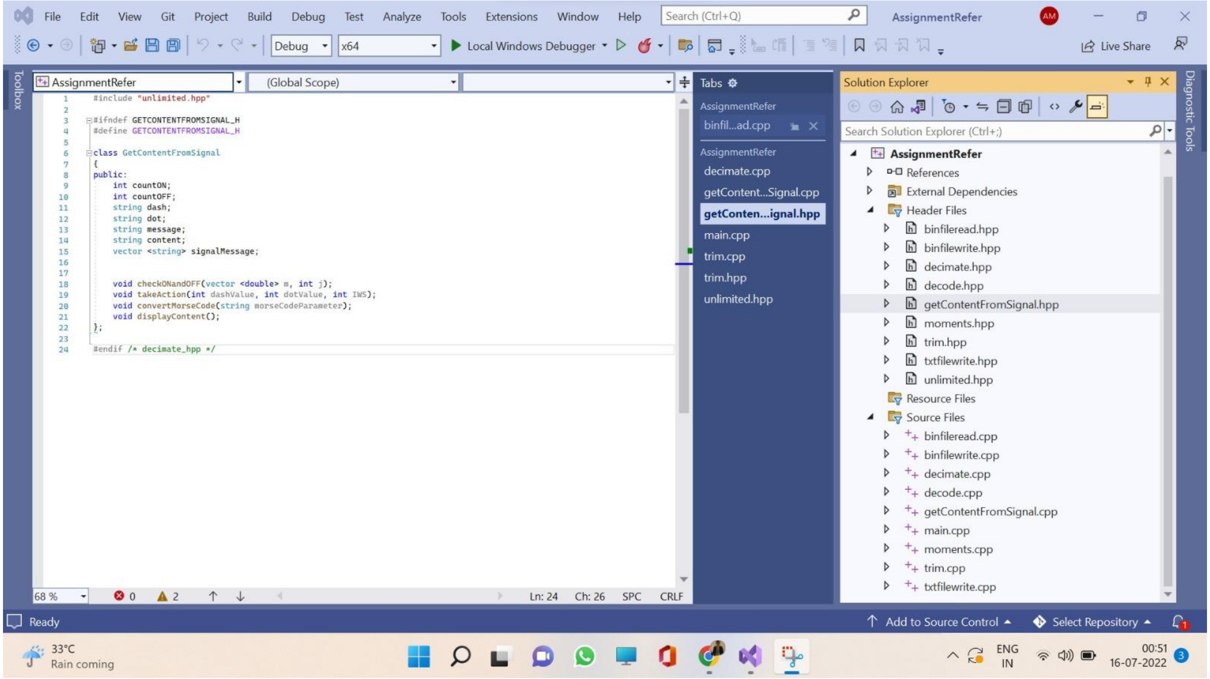


```cpp
    else if (countOFF == dotValue)                   // if signal is OFF for 3 consecutives blocks
    {
        // cout << " " << endl;
        // message += "";
        // cout << "It is IES (Space between two components)" <<endl;
        // cout <<  "" << endl;
    }

    else if (countOFF == dashValue)                   // if signal is OFF for 9 consecutives blocks
    {
        // cout << "It is ICS (Space between two letters)" <<endl;
        message = trim(message);                             //calling the trim function
        // cout << "Message with trimming: " << message << endl;

        for (int i = 0; i < message.length(); i++) // using FOR loop
        {
            char messagefront = message[i];         //getting the first letter of the word
            std::string s(1, messagefront);         //making it string from char
            char messagefronttwo = message[i + 1];  //getting the second letter of the word
            std::string stwo(1, messagefronttwo);   //making it string from char
            string firstTwoCharacter = s + stwo;    //concatenating first and second letter of the word
            // cout << firstTwoCharacter << endl;

            if (firstTwoCharacter == ".-")          //if first and second letter is ".-"
            {
                message.erase(i, 1);            //then erase the first letter
            }
        }
        // cout << "Message after erase: " <<message<< endl;
        convertMorseCode(message);              /* calling the convertMorseCode function to get letter to
                produce word */
                // cout << content << endl;
        message = "";                           /* Once all the above commands are done, empty the message
                variable for holding another new letter and word */
                // cout <<  "" << endl;
    }
    else if (countOFF == IWS)                                 // if the signal is OFF for 21 blocks
    {
        // cout << "It is IWS (Space between two words)" << endl;
        signalMessage.push_back(" ");                         //add a space for having a space between words
        message += "   ";
        // cout <<  "" << endl;
        content += " ";
    }

```



```cpp
}

void GetContentFromSignal::convertMorseCode(string morseCodeParameter)
{

    // created a list for having all Morse Code for different number and letter
    std::list <Morse> listofMorse = { Morse("A", ".-"), Morse("B", "-..."), Morse("C", "-.-.") , Morse("D", "-.."),
                                      Morse("E", "."), Morse("F", "..-."), Morse("G", "--."), Morse("H", "...."),
                                      Morse("I", ".."), Morse("J", ".---"), Morse("K", "-.-"), Morse("L", ".-.."),
                                      Morse("M", "--"), Morse("N", "-."), Morse("O", "---"), Morse("P", ".--."),
                                      Morse("Q", "--.-"), Morse("R", ".-."), Morse("S", "..."), Morse("T", "-"),
                                      Morse("U", "..-"), Morse("V", "...-"), Morse("W", ".--"), Morse("X", "-..-"),
                                      Morse("Y", "-.--"), Morse("Z", "--.."), Morse("1", ".----"), Morse("2", "..---"),
                                      Morse("3", "...--"),Morse("4", "....-"), Morse("5", "....."), Morse("6", "-...."),
                                      Morse("7", "--..."), Morse("8", "---.."), Morse("9", "----."), Morse("0", "-----") };

    //
    std::list<Morse>::iterator it;
    for (it = listofMorse.begin(); it != listofMorse.end(); it++)
    {
        // Accessing the object through iterator
        string alphabet = it->alphabet;
        string morseCode = it->morseCode;
        if (morseCode == morseCodeParameter) // checking if the parameter that is passed match with our stored morse code
        {
            // cout << alphabet <<endl;
            signalMessage.push_back(alphabet); // if there is a match, add it in the vector called signalMessage

        }

    }

}

void GetContentFromSignal::displayContent()
{
    cout << " " << endl;
    cout << "The content of the boris_Nov13_20wpm_short.bin file is: " << endl;
    cout << " " << endl;
    for (auto ac = signalMessage.begin(); ac != signalMessage.end(); ++ac)
        cout << *ac << "";                                              // displaying our content
        // cout << "\e[3m" << *ac << "";                                // comment out to see the content as a BOLD
    cout << " " << endl;
    cout << " " << endl;
}
```

getContentFromSignal.hpp File:



```cpp
#include "unlimited.hpp"

#ifndef GETCONTENTFROMSIGNAL_H
#define GETCONTENTFROMSIGNAL_H

class GetContentFromSignal
{
public:
    int countON;
    int countOFF;
    string dash;
    string dot;
    string message;
    string content;
    vector <string> signalMessage;


    void checkONandOFF(vector <double> m, int j);
    void takeAction(int dashValue, int dotValue, int IWS);
    void convertMorseCode(string morseCodeParameter);
    void displayContent();
};

#endif /* decimate_hpp */
```

Decimate.cpp:

This file is included in the decimate function using the code line:

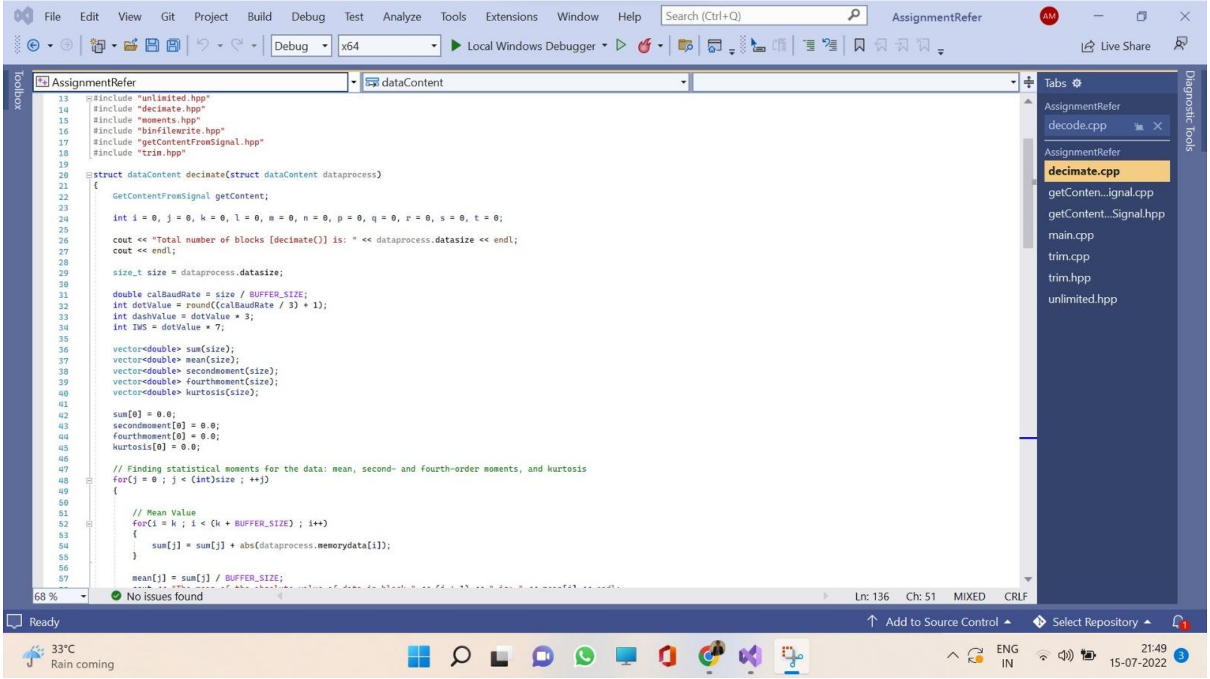
#include “getContentFromSignal.hpp”

So, as the data retrieved can be worked upon. decimate.cpp is then again modified and this getContentFromSignal is called in the decimate.cpp sourcefile.

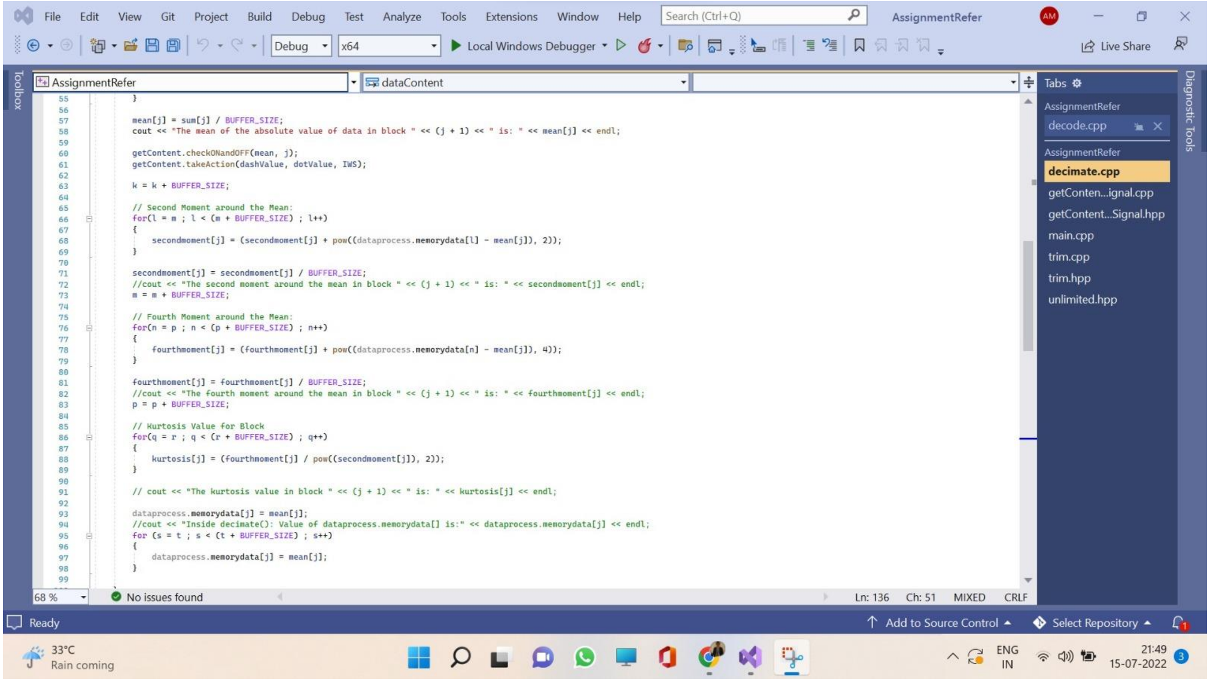
After the successful execution of this file, the program can:

- read the discrete signals
- convert to ON & OFF using the if/else
- convert the combination 1s and 0s to dits and dahs
- then convert the dits and dahs to English alphabets using the listofMorse & iterator.

The final code for decimate file would be:



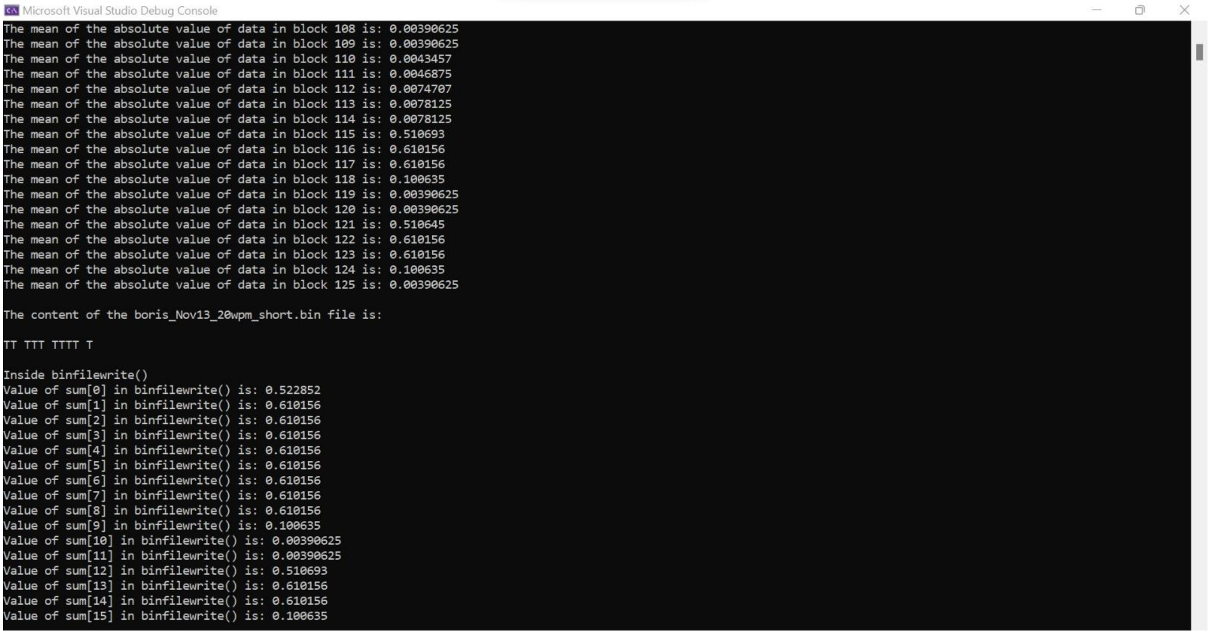
```cpp
#include "unlimited.hpp"
#include "decimate.hpp"
#include "moments.hpp"
#include "binfilewrite.hpp"
#include "getContentFromSignal.hpp"
#include "trim.hpp"

struct dataContent decimate(struct dataContent dataprocess)
{
    GetContentFromSignal getContent;

    int i = 0, j = 0, k = 0, l = 0, m = 0, n = 0, p = 0, q = 0, r = 0, s = 0, t = 0;

    cout << "Total number of blocks [decimate()] is: " << dataprocess.datasize << endl;
    cout << endl;

    size_t size = dataprocess.datasize;

    double calBaudRate = size / BUFFER_SIZE;
    int dotValue = round((calBaudRate / 3) + 1);
    int dashValue = dotValue * 3;
    int IWS = dotValue * 7;

    vector<double> sum(size);
    vector<double> mean(size);
    vector<double> secondmoment(size);
    vector<double> fourthmoment(size);
    vector<double> kurtosis(size);

    sum[0] = 0.0;
    secondmoment[0] = 0.0;
    fourthmoment[0] = 0.0;
    kurtosis[0] = 0.0;

    // Finding statistical moments for the data: mean, second- and fourth-order moments, and kurtosis
    for(j = 0 ; j < (int)size ; ++j)
    {

        // Mean Value
        for(i = k ; i < (k + BUFFER_SIZE) ; i++)
        {
            sum[j] = sum[j] + abs(dataprocess.memorydata[i]);
        }

        mean[j] = sum[j] / BUFFER_SIZE;
```



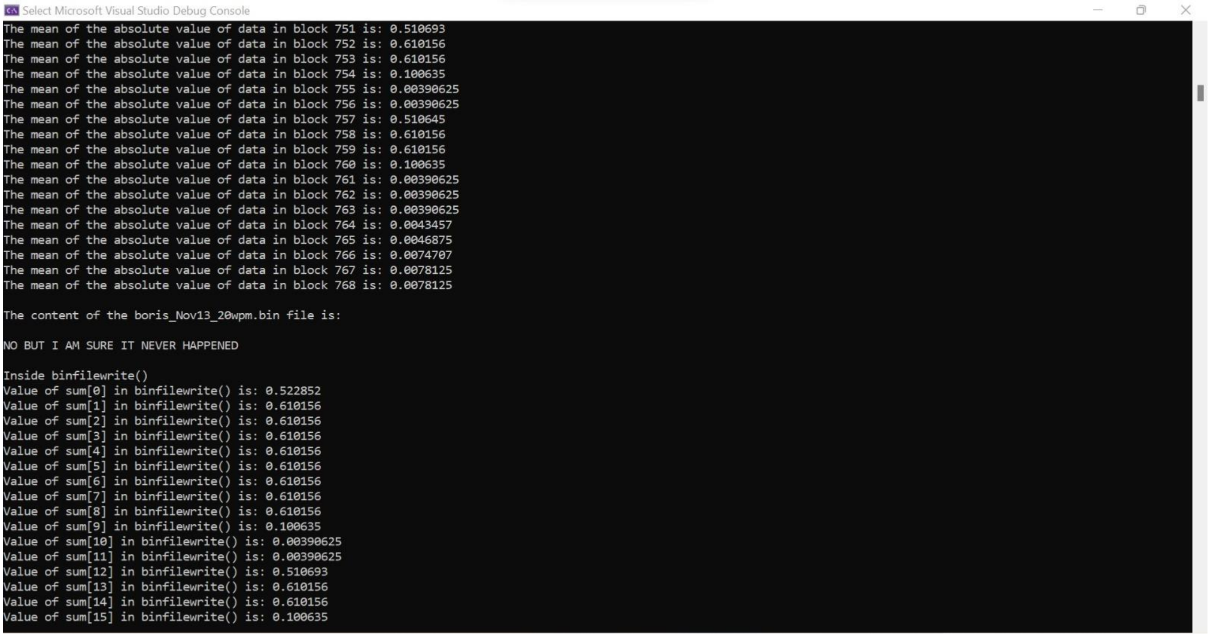
```cpp
        }

        mean[j] = sum[j] / BUFFER_SIZE;
        cout << "The mean of the absolute value of data in block " << (j + 1) << " is: " << mean[j] << endl;

        getContent.checkONandOFF(mean, j);
        getContent.takeAction(dashValue, dotValue, IWS);

        k = k + BUFFER_SIZE;

        // Second Moment around the Mean:
        for(l = m ; l < (m + BUFFER_SIZE) ; l++)
        {
            secondmoment[j] = (secondmoment[j] + pow((dataprocess.memorydata[l] - mean[j]), 2));
        }

        secondmoment[j] = secondmoment[j] / BUFFER_SIZE;
        //cout << "The second moment around the mean in block " << (j + 1) << " is: " << secondmoment[j] << endl;
        m = m + BUFFER_SIZE;

        // Fourth Moment around the Mean:
        for(n = p ; n < (p + BUFFER_SIZE) ; n++)
        {
            fourthmoment[j] = (fourthmoment[j] + pow((dataprocess.memorydata[n] - mean[j]), 4));
        }

        fourthmoment[j] = fourthmoment[j] / BUFFER_SIZE;
        //cout << "The fourth moment around the mean in block " << (j + 1) << " is: " << fourthmoment[j] << endl;
        p = p + BUFFER_SIZE;

        // Kurtosis Value for Block
        for(q = r ; q < (r + BUFFER_SIZE) ; q++)
        {
            kurtosis[j] = (fourthmoment[j] / pow((secondmoment[j]), 2));
        }

        // cout << "The kurtosis value in block " << (j + 1) << " is: " << kurtosis[j] << endl;

        dataprocess.memorydata[j] = mean[j];
        //cout << "Inside decimate(): Value of dataprocess.memorydata[] is:" << dataprocess.memorydata[j] << endl;
        for (s = t ; s < (t + BUFFER_SIZE) ; s++)
        {
            dataprocess.memorydata[j] = mean[j];
        }

```

Final Outputs of the binary files:

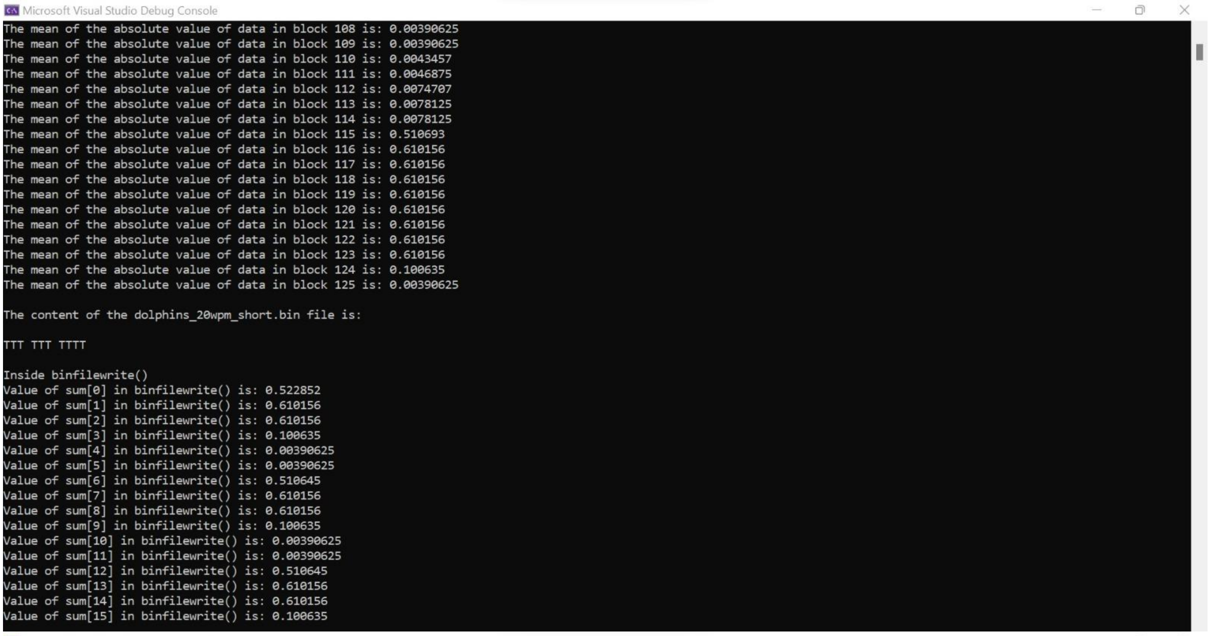
1. dolphins_20wpm.bin
2. dolphins_20wpm_short.bin
3. boris_Nov13_20wpm.bin
4. boris_Nov13_20wpm_short.bin
5. dolphins_10wpm.bin
6. dolphins_10wpm_short.bin



```
The mean of the absolute value of data in block 108 is: 0.00390625
The mean of the absolute value of data in block 109 is: 0.00390625
The mean of the absolute value of data in block 110 is: 0.0043457
The mean of the absolute value of data in block 111 is: 0.0046875
The mean of the absolute value of data in block 112 is: 0.0074707
The mean of the absolute value of data in block 113 is: 0.0078125
The mean of the absolute value of data in block 114 is: 0.0078125
The mean of the absolute value of data in block 115 is: 0.510693
The mean of the absolute value of data in block 116 is: 0.610156
The mean of the absolute value of data in block 117 is: 0.610156
The mean of the absolute value of data in block 118 is: 0.100635
The mean of the absolute value of data in block 119 is: 0.00390625
The mean of the absolute value of data in block 120 is: 0.00390625
The mean of the absolute value of data in block 121 is: 0.510645
The mean of the absolute value of data in block 122 is: 0.610156
The mean of the absolute value of data in block 123 is: 0.610156
The mean of the absolute value of data in block 124 is: 0.100635
The mean of the absolute value of data in block 125 is: 0.00390625

The content of the boris_Nov13_20wpm_short.bin file is:

TT TTT TTTT T

Inside binfilewrite()
Value of sum[0] in binfilewrite() is: 0.522852
Value of sum[1] in binfilewrite() is: 0.610156
Value of sum[2] in binfilewrite() is: 0.610156
Value of sum[3] in binfilewrite() is: 0.610156
Value of sum[4] in binfilewrite() is: 0.610156
Value of sum[5] in binfilewrite() is: 0.610156
Value of sum[6] in binfilewrite() is: 0.610156
Value of sum[7] in binfilewrite() is: 0.610156
Value of sum[8] in binfilewrite() is: 0.610156
Value of sum[9] in binfilewrite() is: 0.100635
Value of sum[10] in binfilewrite() is: 0.00390625
Value of sum[11] in binfilewrite() is: 0.00390625
Value of sum[12] in binfilewrite() is: 0.510693
Value of sum[13] in binfilewrite() is: 0.610156
Value of sum[14] in binfilewrite() is: 0.610156
Value of sum[15] in binfilewrite() is: 0.100635
```

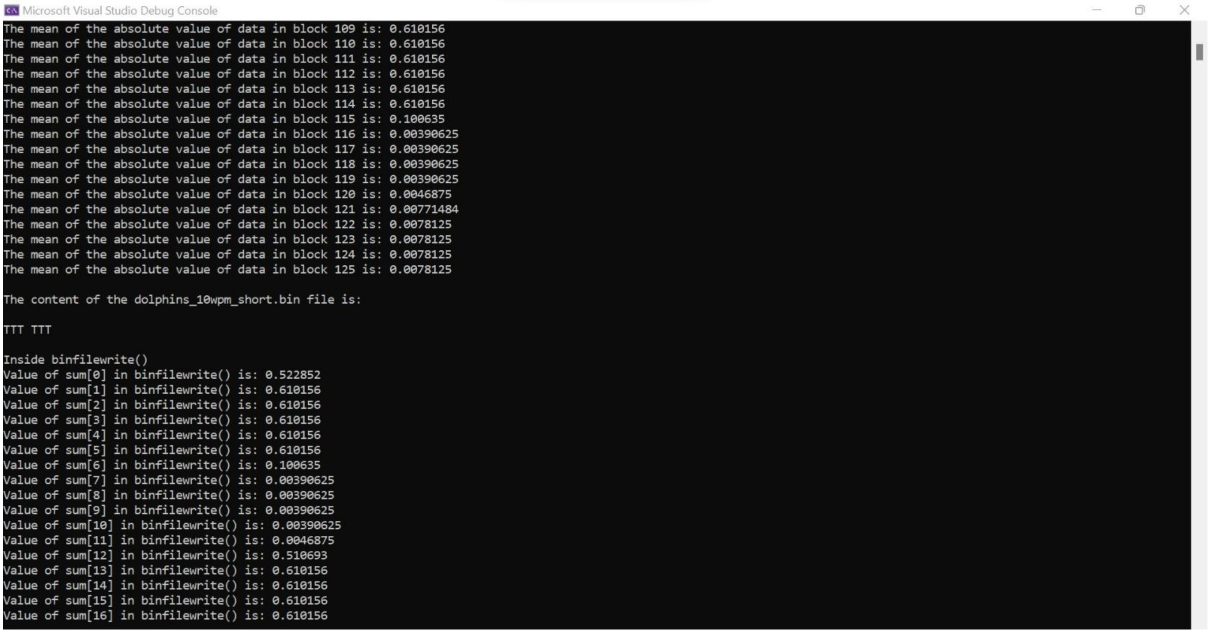


```
The mean of the absolute value of data in block 751 is: 0.510693
The mean of the absolute value of data in block 752 is: 0.610156
The mean of the absolute value of data in block 753 is: 0.610156
The mean of the absolute value of data in block 754 is: 0.100635
The mean of the absolute value of data in block 755 is: 0.00390625
The mean of the absolute value of data in block 756 is: 0.00390625
The mean of the absolute value of data in block 757 is: 0.510645
The mean of the absolute value of data in block 758 is: 0.610156
The mean of the absolute value of data in block 759 is: 0.610156
The mean of the absolute value of data in block 760 is: 0.100635
The mean of the absolute value of data in block 761 is: 0.00390625
The mean of the absolute value of data in block 762 is: 0.00390625
The mean of the absolute value of data in block 763 is: 0.00390625
The mean of the absolute value of data in block 764 is: 0.0043457
The mean of the absolute value of data in block 765 is: 0.0046875
The mean of the absolute value of data in block 766 is: 0.0074707
The mean of the absolute value of data in block 767 is: 0.0078125
The mean of the absolute value of data in block 768 is: 0.0078125

The content of the boris_Nov13_20wpm.bin file is:

NO BUT I AM SURE IT NEVER HAPPENED

Inside binfilewrite()
Value of sum[0] in binfilewrite() is: 0.522852
Value of sum[1] in binfilewrite() is: 0.610156
Value of sum[2] in binfilewrite() is: 0.610156
Value of sum[3] in binfilewrite() is: 0.610156
Value of sum[4] in binfilewrite() is: 0.610156
Value of sum[5] in binfilewrite() is: 0.610156
Value of sum[6] in binfilewrite() is: 0.610156
Value of sum[7] in binfilewrite() is: 0.610156
Value of sum[8] in binfilewrite() is: 0.610156
Value of sum[9] in binfilewrite() is: 0.100635
Value of sum[10] in binfilewrite() is: 0.00390625
Value of sum[11] in binfilewrite() is: 0.00390625
Value of sum[12] in binfilewrite() is: 0.510693
Value of sum[13] in binfilewrite() is: 0.610156
Value of sum[14] in binfilewrite() is: 0.610156
Value of sum[15] in binfilewrite() is: 0.100635
```



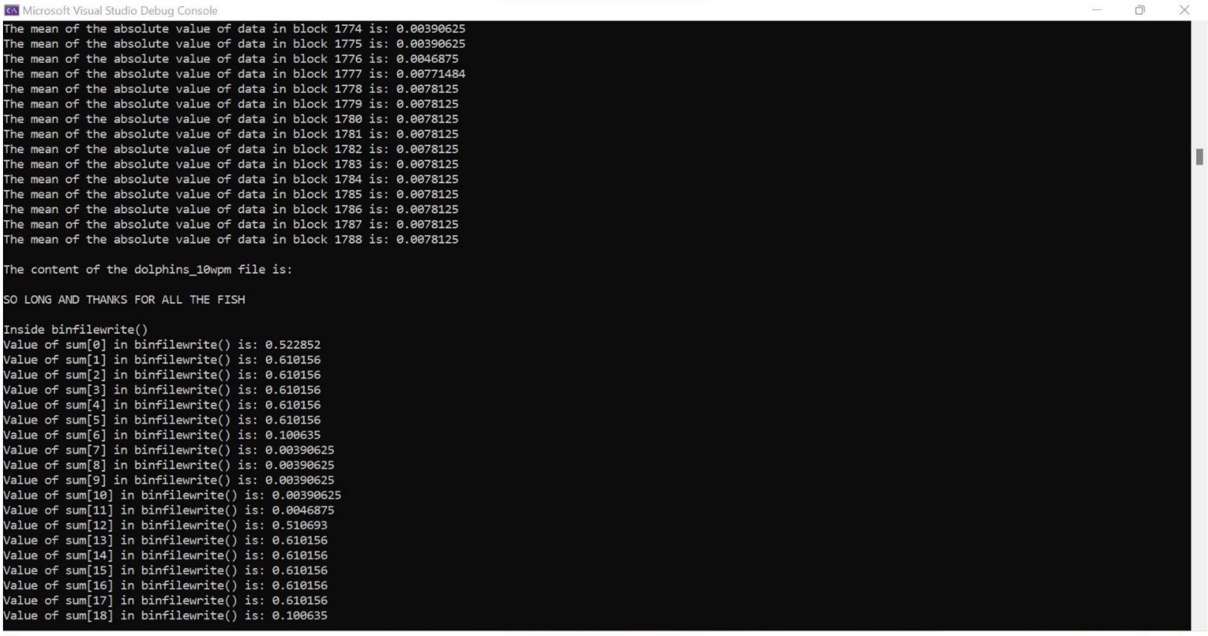
Microsoft Visual Studio Debug Console

```
The mean of the absolute value of data in block 108 is: 0.00390625
The mean of the absolute value of data in block 109 is: 0.00390625
The mean of the absolute value of data in block 110 is: 0.0043457
The mean of the absolute value of data in block 111 is: 0.0046875
The mean of the absolute value of data in block 112 is: 0.0074707
The mean of the absolute value of data in block 113 is: 0.0078125
The mean of the absolute value of data in block 114 is: 0.0078125
The mean of the absolute value of data in block 115 is: 0.510693
The mean of the absolute value of data in block 116 is: 0.610156
The mean of the absolute value of data in block 117 is: 0.610156
The mean of the absolute value of data in block 118 is: 0.610156
The mean of the absolute value of data in block 119 is: 0.610156
The mean of the absolute value of data in block 120 is: 0.610156
The mean of the absolute value of data in block 121 is: 0.610156
The mean of the absolute value of data in block 122 is: 0.610156
The mean of the absolute value of data in block 123 is: 0.610156
The mean of the absolute value of data in block 124 is: 0.100635
The mean of the absolute value of data in block 125 is: 0.00390625

The content of the dolphins_20wpm_short.bin file is:

TTT TTT TTTT

Inside binfilewrite()
Value of sum[0] in binfilewrite() is: 0.522852
Value of sum[1] in binfilewrite() is: 0.610156
Value of sum[2] in binfilewrite() is: 0.610156
Value of sum[3] in binfilewrite() is: 0.100635
Value of sum[4] in binfilewrite() is: 0.00390625
Value of sum[5] in binfilewrite() is: 0.00390625
Value of sum[6] in binfilewrite() is: 0.510645
Value of sum[7] in binfilewrite() is: 0.610156
Value of sum[8] in binfilewrite() is: 0.610156
Value of sum[9] in binfilewrite() is: 0.100635
Value of sum[10] in binfilewrite() is: 0.00390625
Value of sum[11] in binfilewrite() is: 0.00390625
Value of sum[12] in binfilewrite() is: 0.510645
Value of sum[13] in binfilewrite() is: 0.610156
Value of sum[14] in binfilewrite() is: 0.610156
Value of sum[15] in binfilewrite() is: 0.100635
```

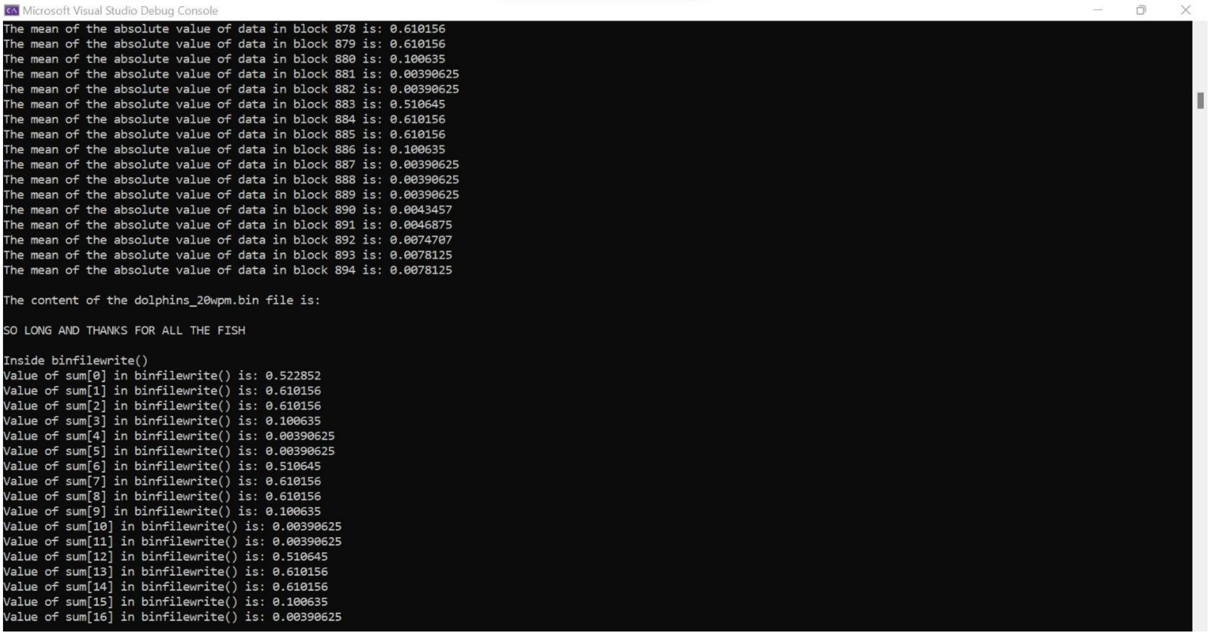


Microsoft Visual Studio Debug Console

```
The mean of the absolute value of data in block 109 is: 0.610156
The mean of the absolute value of data in block 110 is: 0.610156
The mean of the absolute value of data in block 111 is: 0.610156
The mean of the absolute value of data in block 112 is: 0.610156
The mean of the absolute value of data in block 113 is: 0.610156
The mean of the absolute value of data in block 114 is: 0.610156
The mean of the absolute value of data in block 115 is: 0.100635
The mean of the absolute value of data in block 116 is: 0.00390625
The mean of the absolute value of data in block 117 is: 0.00390625
The mean of the absolute value of data in block 118 is: 0.00390625
The mean of the absolute value of data in block 119 is: 0.00390625
The mean of the absolute value of data in block 120 is: 0.0046875
The mean of the absolute value of data in block 121 is: 0.00771484
The mean of the absolute value of data in block 122 is: 0.0078125
The mean of the absolute value of data in block 123 is: 0.0078125
The mean of the absolute value of data in block 124 is: 0.0078125
The mean of the absolute value of data in block 125 is: 0.0078125

The content of the dolphins_10wpm_short.bin file is:

TTT TTT

Inside binfilewrite()
Value of sum[0] in binfilewrite() is: 0.522852
Value of sum[1] in binfilewrite() is: 0.610156
Value of sum[2] in binfilewrite() is: 0.610156
Value of sum[3] in binfilewrite() is: 0.610156
Value of sum[4] in binfilewrite() is: 0.610156
Value of sum[5] in binfilewrite() is: 0.610156
Value of sum[6] in binfilewrite() is: 0.100635
Value of sum[7] in binfilewrite() is: 0.00390625
Value of sum[8] in binfilewrite() is: 0.00390625
Value of sum[9] in binfilewrite() is: 0.00390625
Value of sum[10] in binfilewrite() is: 0.00390625
Value of sum[11] in binfilewrite() is: 0.0046875
Value of sum[12] in binfilewrite() is: 0.510693
Value of sum[13] in binfilewrite() is: 0.610156
Value of sum[14] in binfilewrite() is: 0.610156
Value of sum[15] in binfilewrite() is: 0.610156
Value of sum[16] in binfilewrite() is: 0.610156
```



```
The mean of the absolute value of data in block 1774 is: 0.00390625
The mean of the absolute value of data in block 1775 is: 0.00390625
The mean of the absolute value of data in block 1776 is: 0.0046875
The mean of the absolute value of data in block 1777 is: 0.00771484
The mean of the absolute value of data in block 1778 is: 0.0078125
The mean of the absolute value of data in block 1779 is: 0.0078125
The mean of the absolute value of data in block 1780 is: 0.0078125
The mean of the absolute value of data in block 1781 is: 0.0078125
The mean of the absolute value of data in block 1782 is: 0.0078125
The mean of the absolute value of data in block 1783 is: 0.0078125
The mean of the absolute value of data in block 1784 is: 0.0078125
The mean of the absolute value of data in block 1785 is: 0.0078125
The mean of the absolute value of data in block 1786 is: 0.0078125
The mean of the absolute value of data in block 1787 is: 0.0078125
The mean of the absolute value of data in block 1788 is: 0.0078125

The content of the dolphins_10wpm file is:

SO LONG AND THANKS FOR ALL THE FISH

Inside binfilewrite()
Value of sum[0] in binfilewrite() is: 0.522852
Value of sum[1] in binfilewrite() is: 0.610156
Value of sum[2] in binfilewrite() is: 0.610156
Value of sum[3] in binfilewrite() is: 0.610156
Value of sum[4] in binfilewrite() is: 0.610156
Value of sum[5] in binfilewrite() is: 0.610156
Value of sum[6] in binfilewrite() is: 0.100635
Value of sum[7] in binfilewrite() is: 0.00390625
Value of sum[8] in binfilewrite() is: 0.00390625
Value of sum[9] in binfilewrite() is: 0.00390625
Value of sum[10] in binfilewrite() is: 0.00390625
Value of sum[11] in binfilewrite() is: 0.0046875
Value of sum[12] in binfilewrite() is: 0.510693
Value of sum[13] in binfilewrite() is: 0.610156
Value of sum[14] in binfilewrite() is: 0.610156
Value of sum[15] in binfilewrite() is: 0.610156
Value of sum[16] in binfilewrite() is: 0.610156
Value of sum[17] in binfilewrite() is: 0.610156
Value of sum[18] in binfilewrite() is: 0.100635
```



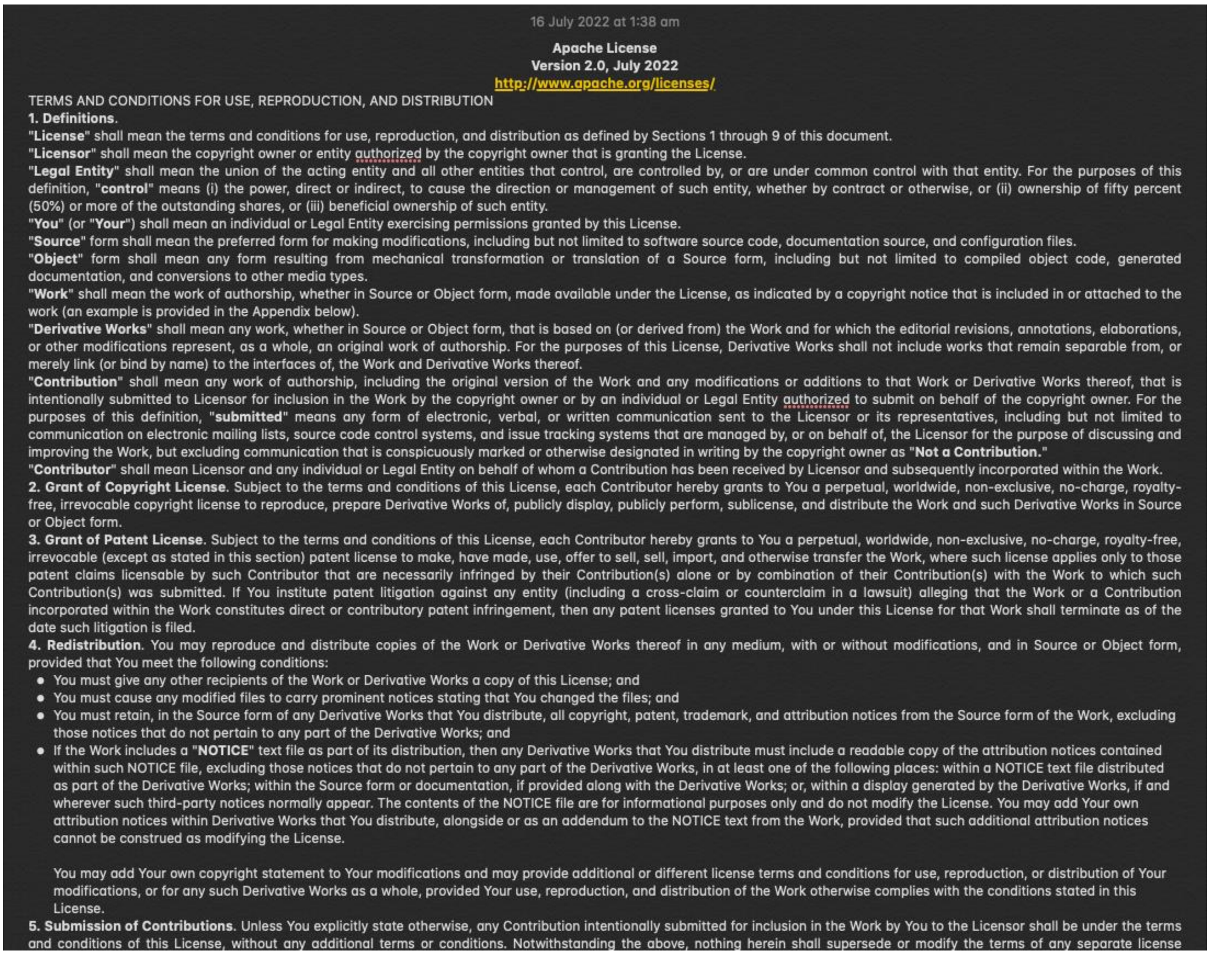
```
The mean of the absolute value of data in block 878 is: 0.610156
The mean of the absolute value of data in block 879 is: 0.610156
The mean of the absolute value of data in block 880 is: 0.100635
The mean of the absolute value of data in block 881 is: 0.00390625
The mean of the absolute value of data in block 882 is: 0.00390625
The mean of the absolute value of data in block 883 is: 0.510645
The mean of the absolute value of data in block 884 is: 0.610156
The mean of the absolute value of data in block 885 is: 0.610156
The mean of the absolute value of data in block 886 is: 0.100635
The mean of the absolute value of data in block 887 is: 0.00390625
The mean of the absolute value of data in block 888 is: 0.00390625
The mean of the absolute value of data in block 889 is: 0.00390625
The mean of the absolute value of data in block 890 is: 0.0043457
The mean of the absolute value of data in block 891 is: 0.0046875
The mean of the absolute value of data in block 892 is: 0.0074707
The mean of the absolute value of data in block 893 is: 0.0078125
The mean of the absolute value of data in block 894 is: 0.0078125

The content of the dolphins_20wpm.bin file is:

SO LONG AND THANKS FOR ALL THE FISH

Inside binfilewrite()
Value of sum[0] in binfilewrite() is: 0.522852
Value of sum[1] in binfilewrite() is: 0.610156
Value of sum[2] in binfilewrite() is: 0.610156
Value of sum[3] in binfilewrite() is: 0.100635
Value of sum[4] in binfilewrite() is: 0.00390625
Value of sum[5] in binfilewrite() is: 0.00390625
Value of sum[6] in binfilewrite() is: 0.510645
Value of sum[7] in binfilewrite() is: 0.610156
Value of sum[8] in binfilewrite() is: 0.610156
Value of sum[9] in binfilewrite() is: 0.100635
Value of sum[10] in binfilewrite() is: 0.00390625
Value of sum[11] in binfilewrite() is: 0.00390625
Value of sum[12] in binfilewrite() is: 0.510645
Value of sum[13] in binfilewrite() is: 0.610156
Value of sum[14] in binfilewrite() is: 0.610156
Value of sum[15] in binfilewrite() is: 0.100635
Value of sum[16] in binfilewrite() is: 0.00390625
```

Task 4:

License and mentioning the terms and conditions for the program used is an essential part of building computer coding.

- They ensure developer's intellectual trade secrets.
- They monitor if any other party is using the code illegally.
- They help users in maintaining a positive relationship between the coder and the vendor.
- They define the clear boundaries of what can be done with the code and what should not be done.

License:



16 July 2022 at 1:38 am

Apache License
Version 2.0, July 2022
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"**License**" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"**Licensor**" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"**Legal Entity**" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "**control**" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"**You**" (or "**Your**") shall mean an individual or Legal Entity exercising permissions granted by this License.

"**Source**" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"**Object**" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"**Work**" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"**Derivative Works**" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"**Contribution**" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "**submitted**" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "**Not a Contribution.**"

"**Contributor**" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

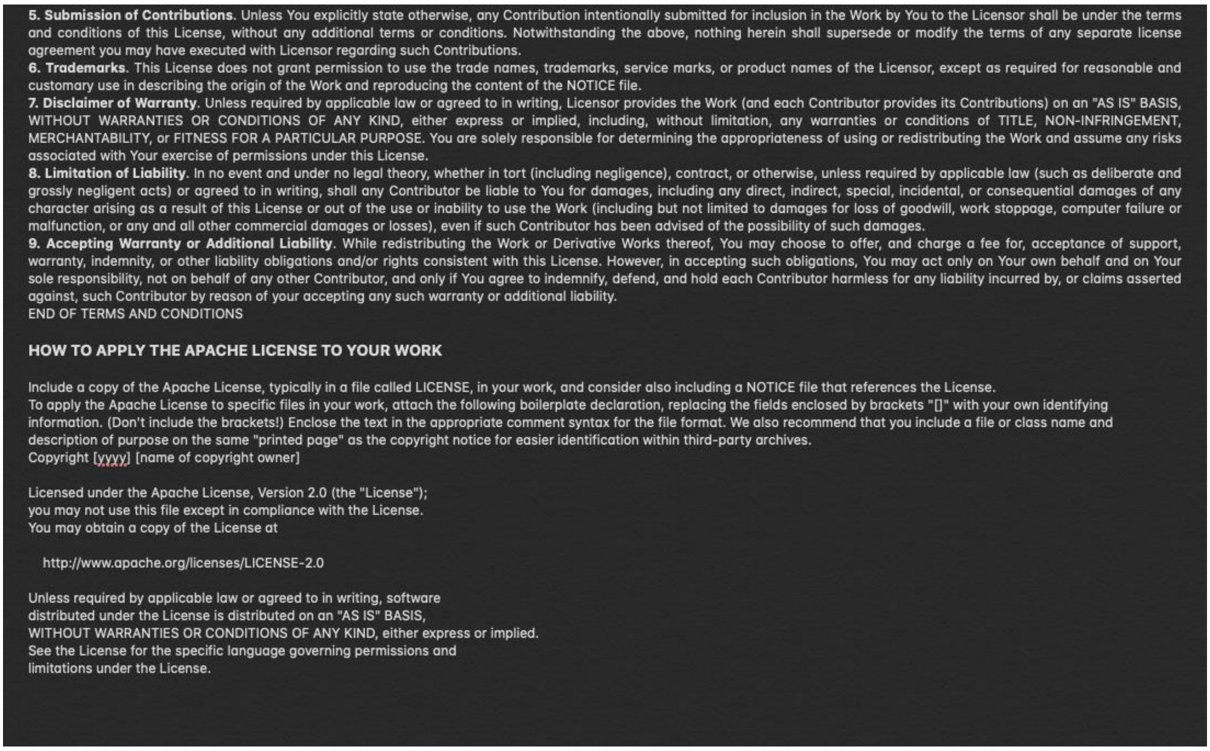
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

- You must give any other recipients of the Work or Derivative Works a copy of this License; and
- You must cause any modified files to carry prominent notices stating that You changed the files; and
- You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
- If the Work includes a "**NOTICE**" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

  You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license



5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

HOW TO APPLY THE APACHE LICENSE TO YOUR WORK

Include a copy of the Apache License, typically in a file called LICENSE, in your work, and consider also including a NOTICE file that references the License.

To apply the Apache License to specific files in your work, attach the following boilerplate declaration, replacing the fields enclosed by brackets "[]" with your own identifying information. (Don't include the brackets!) Enclose the text in the appropriate comment syntax for the file format. We also recommend that you include a file or class name and description of purpose on the same "printed page" as the copyright notice for easier identification within third-party archives.

```
Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

   http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```