CSE-598 MIS Project Phase-2 (Group 8)

Karen Tamayo, Karthik Vivekanandan, Marcela Tejo, Rahul Krishna Vasantham, Yuhan Sun.

Task 1

what?

First Task is to write a Temporal Encoding function to encode the given video
Step 1: Get the frames in the given video file
Step 2: Process the frames by getting a part of the frame (based on the input provided by the user with size fixed to 10*10)
Step 3: Get the Y component of every pixel from all the frames in the video and sort them out in temporal(time) basis.
Step 4: Ask the user for the choice of his/her predictor function to encode the information.
Step 5: Once the output is compute output the same to a file with the format X_Y.tpc.
(X -> video file name, Y -> Option chosen for predictor function)
Step 6: Output the absolute Error (absolute sum of all the non-absolute individual error) to the terminal

How?
Step 1: Once the input choice of video is taken from the user, we used [*cv2.VideoCapture ()]* to go through the video and extract each frame.
Step 2: Basing on the width and height start points chosen by the user, we get a 10*10 image portion from each frame using [*frame[height:height+row_length, width:width+col_length]]*
Step 3: While extracting each frame we create a signal list with 100 rows and columns equal to the number of frames, where each row will have all the pixels of a particular index from all the frames in one row. so we have a total of 100 rows.
Step 4: Encoding the data:
Now no encoding option (option -1) is chosen by the user exact pixel information is written to the file
If option - 2,3,4 is chosen, first pixel is written as it is and from the second pixel onwards, we compute the expected value using the predictive function [*S[t] = S[t-1]]* if there is any change in the expected value from the actual value, we find the error, (actual value - predicted value) and store the values.
In case of option-3, first two values are left unchanged and the error values are calculated from third value as we need 2 values for prediction.
In case of option-4, we have taken the conditions

$\alpha_1 + \alpha_2 = 1.0$ and

$$\alpha_1 \times s_i[t-2]+\alpha_2 \times s_i[t-3]=s_i[\ -1]$$
$$\alpha_1 \times s_i[t-3]+\alpha_2 \times s_i[t-4]=s_i[t-2]$$

into consideration to compute the alpha1 and alpha2 values.

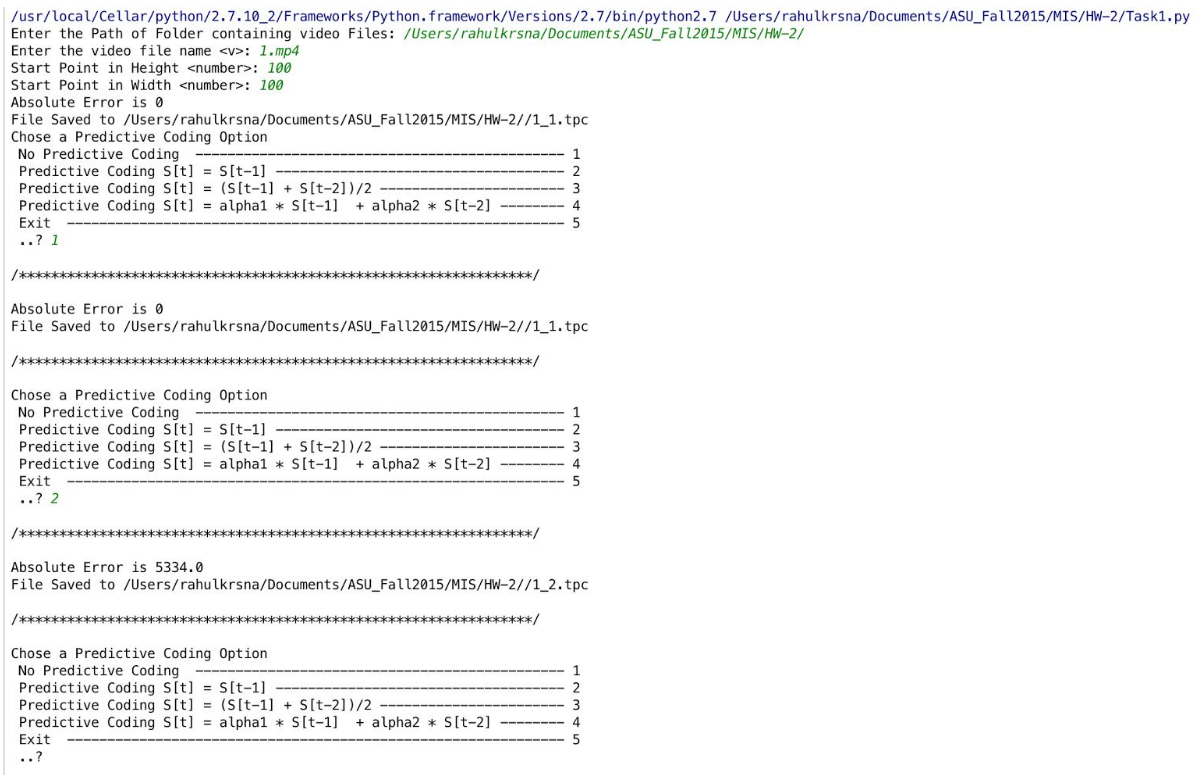
In this case, first two values are given as it is, for the third value, I have chosen the values of alpha1 and alpha2 to be 0.5 as I cannot compute it directly here. Also any place while computing alpha1 and alpha2 have violated the rules of >=0 and sum to be 1, I have considered them to be 0.5

Step 5: Finally the errors computed are saved to the file in the X_Y.spc file.

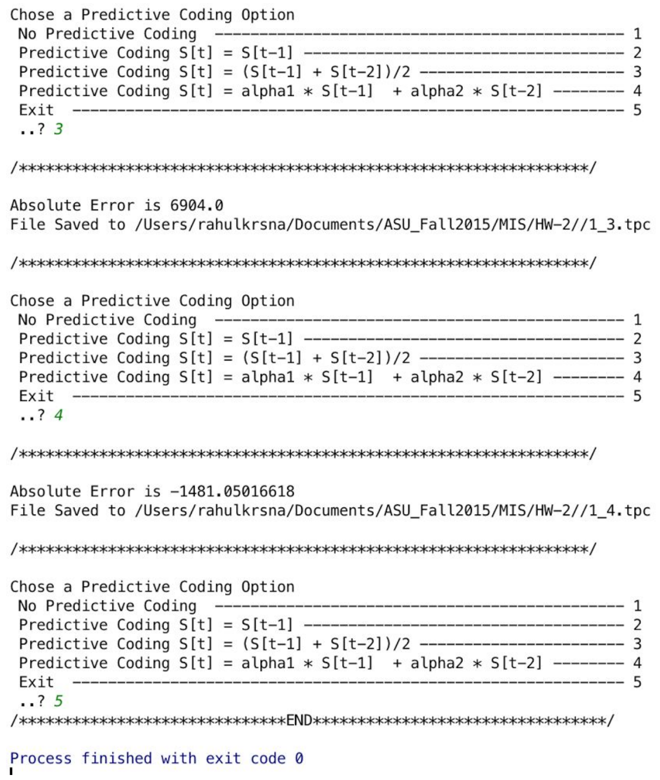
Step 6: At every step, while computing the errors, we sum all the errors and finally the absolute value of the errors is printed out to the user.

Output

Task - 1



```
/usr/local/Cellar/python/2.7.10_2/Frameworks/Python.framework/Versions/2.7/bin/python2.7 /Users/rahulkrsna/Documents/ASU_Fall2015/MIS/HW-2/Task1.py
Enter the Path of Folder containing video Files: /Users/rahulkrsna/Documents/ASU_Fall2015/MIS/HW-2/
Enter the video file name <v>: 1.mp4
Start Point in Height <number>: 100
Start Point in Width <number>: 100
Absolute Error is 0
File Saved to /Users/rahulkrsna/Documents/ASU_Fall2015/MIS/HW-2//1_1.tpc
Chose a Predictive Coding Option
 No Predictive Coding  ----------------------------------------------- 1
 Predictive Coding S[t] = S[t-1] ------------------------------------- 2
 Predictive Coding S[t] = (S[t-1] + S[t-2])/2 ----------------------- 3
 Predictive Coding S[t] = alpha1 * S[t-1]  + alpha2 * S[t-2] -------- 4
 Exit  -------------------------------------------------------------- 5
 ..? 1

/****************************************************************/

Absolute Error is 0
File Saved to /Users/rahulkrsna/Documents/ASU_Fall2015/MIS/HW-2//1_1.tpc

/****************************************************************/

Chose a Predictive Coding Option
 No Predictive Coding  ----------------------------------------------- 1
 Predictive Coding S[t] = S[t-1] ------------------------------------- 2
 Predictive Coding S[t] = (S[t-1] + S[t-2])/2 ----------------------- 3
 Predictive Coding S[t] = alpha1 * S[t-1]  + alpha2 * S[t-2] -------- 4
 Exit  -------------------------------------------------------------- 5
 ..? 2

/****************************************************************/

Absolute Error is 5334.0
File Saved to /Users/rahulkrsna/Documents/ASU_Fall2015/MIS/HW-2//1_2.tpc

/****************************************************************/

Chose a Predictive Coding Option
 No Predictive Coding  ----------------------------------------------- 1
 Predictive Coding S[t] = S[t-1] ------------------------------------- 2
 Predictive Coding S[t] = (S[t-1] + S[t-2])/2 ----------------------- 3
 Predictive Coding S[t] = alpha1 * S[t-1]  + alpha2 * S[t-2] -------- 4
 Exit  -------------------------------------------------------------- 5
 ..?
```



```
Chose a Predictive Coding Option
 No Predictive Coding  ---------------------------------------------- 1
 Predictive Coding S[t] = S[t-1] ------------------------------------ 2
 Predictive Coding S[t] = (S[t-1] + S[t-2])/2 ----------------------- 3
 Predictive Coding S[t] = alpha1 * S[t-1]  + alpha2 * S[t-2] -------- 4
 Exit  -------------------------------------------------------------- 5
 ..? 3

/****************************************************************/

Absolute Error is 6904.0
File Saved to /Users/rahulkrsna/Documents/ASU_Fall2015/MIS/HW-2//1_3.tpc

/****************************************************************/

Chose a Predictive Coding Option
 No Predictive Coding  ---------------------------------------------- 1
 Predictive Coding S[t] = S[t-1] ------------------------------------ 2
 Predictive Coding S[t] = (S[t-1] + S[t-2])/2 ----------------------- 3
 Predictive Coding S[t] = alpha1 * S[t-1]  + alpha2 * S[t-2] -------- 4
 Exit  -------------------------------------------------------------- 5
 ..? 4

/****************************************************************/

Absolute Error is -1481.05016618
File Saved to /Users/rahulkrsna/Documents/ASU_Fall2015/MIS/HW-2//1_4.tpc

/****************************************************************/

Chose a Predictive Coding Option
 No Predictive Coding  ---------------------------------------------- 1
 Predictive Coding S[t] = S[t-1] ------------------------------------ 2
 Predictive Coding S[t] = (S[t-1] + S[t-2])/2 ----------------------- 3
 Predictive Coding S[t] = alpha1 * S[t-1]  + alpha2 * S[t-2] -------- 4
 Exit  -------------------------------------------------------------- 5
 ..? 5
/*******************************END*********************************/

Process finished with exit code 0
```

The output is written to the files as a list of numbers.

Task 2

Task 2 use different predictor to do the coding and output result into a text file. Additionally, the total absolute prediction error will printed on screen.

Given by a user the video file name, we first utilize the opencv function VideoCapture() to get the video. Based on the given top-left X and top-left Y coordinate, truncate the video into the exact region with shape 10*10. Then we use our own function Coding() to do the coding work.

In Coding(), for each choice, we do different job frame by frame.

For predictor option 1, we use exact the original values.

For predictor option 2, we use predictor A which is the left-side value. So we need to keep the first column of each frame. For each value in other columns, we store only the difference between itself and its left-side value.

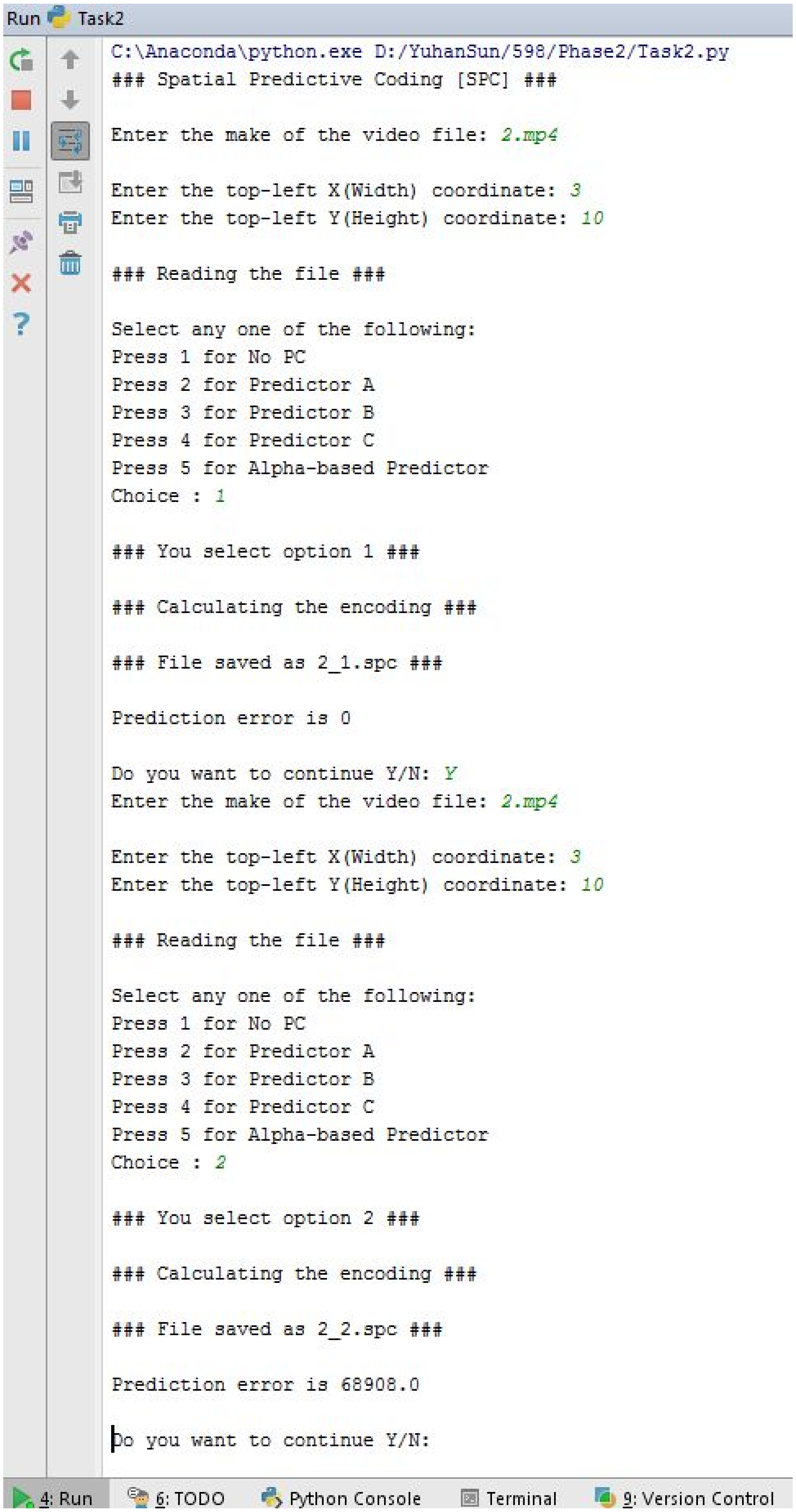
For predictor option 3, we use predictor B which is the upper value. So we keep the first row of each frame. For each value other rows, we store only the difference between itself and its upper value.

For predictor option 4, we use predictor C which is the left-upper value. So we need to keep not only the first row but also the first column. For values in other position in the frame, we only store its difference to its left-upper value.

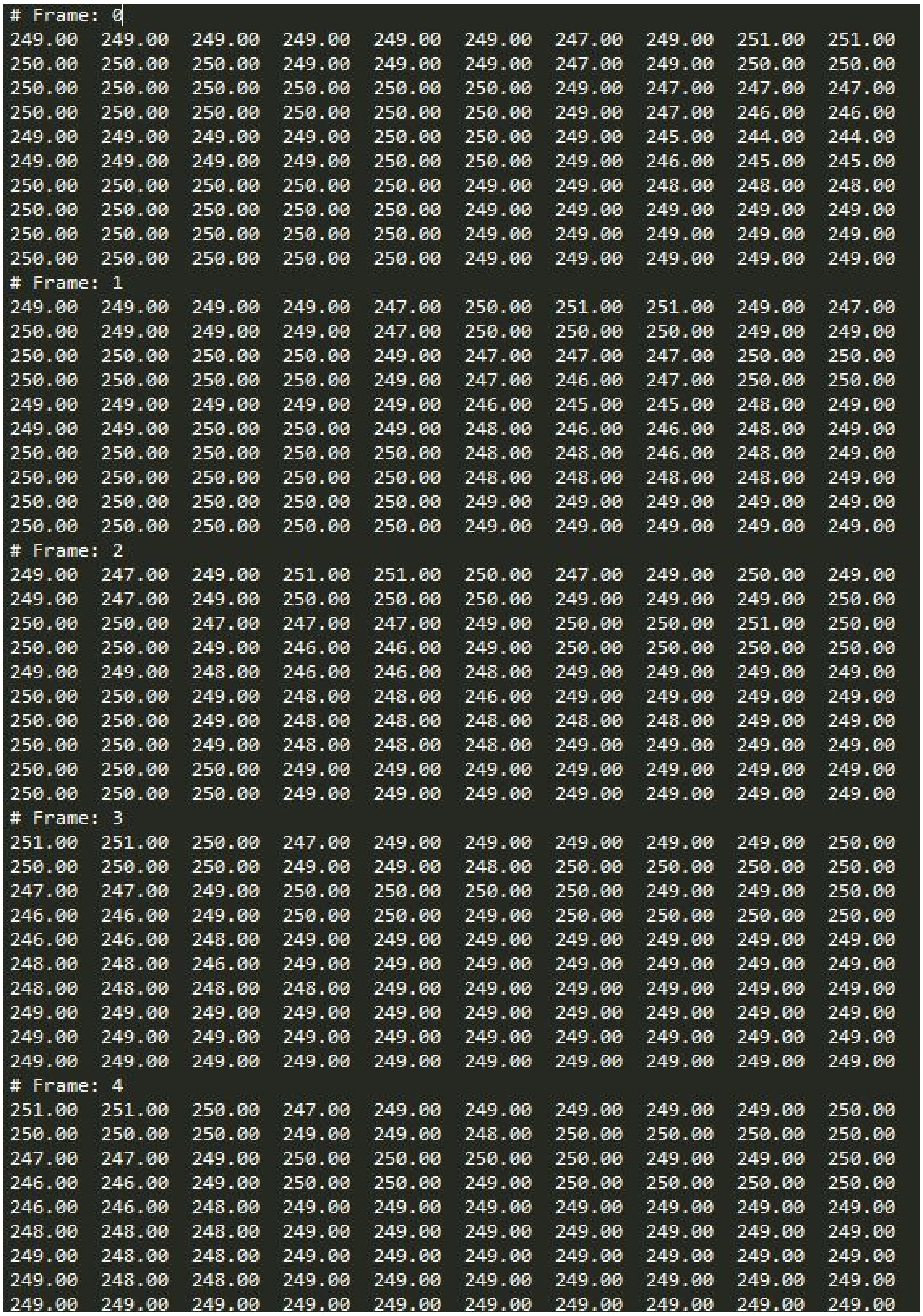
For predictor option 5, we use combination of A, B, C using equation $\frac{1}{3}(A+B+C)$ as the predictor value for each cell. A, B, C is the same relative position to what we've already mentioned. We still need to store exactly values in the first row and the first column. For other values, we store the difference between its value and $\frac{1}{3}(A+B+C)$.

Error of predictor A,B,C and $\frac{1}{3}(A+B+C)$ can be accumulated when we calculate the differences. Then the error will be printed on the screen.

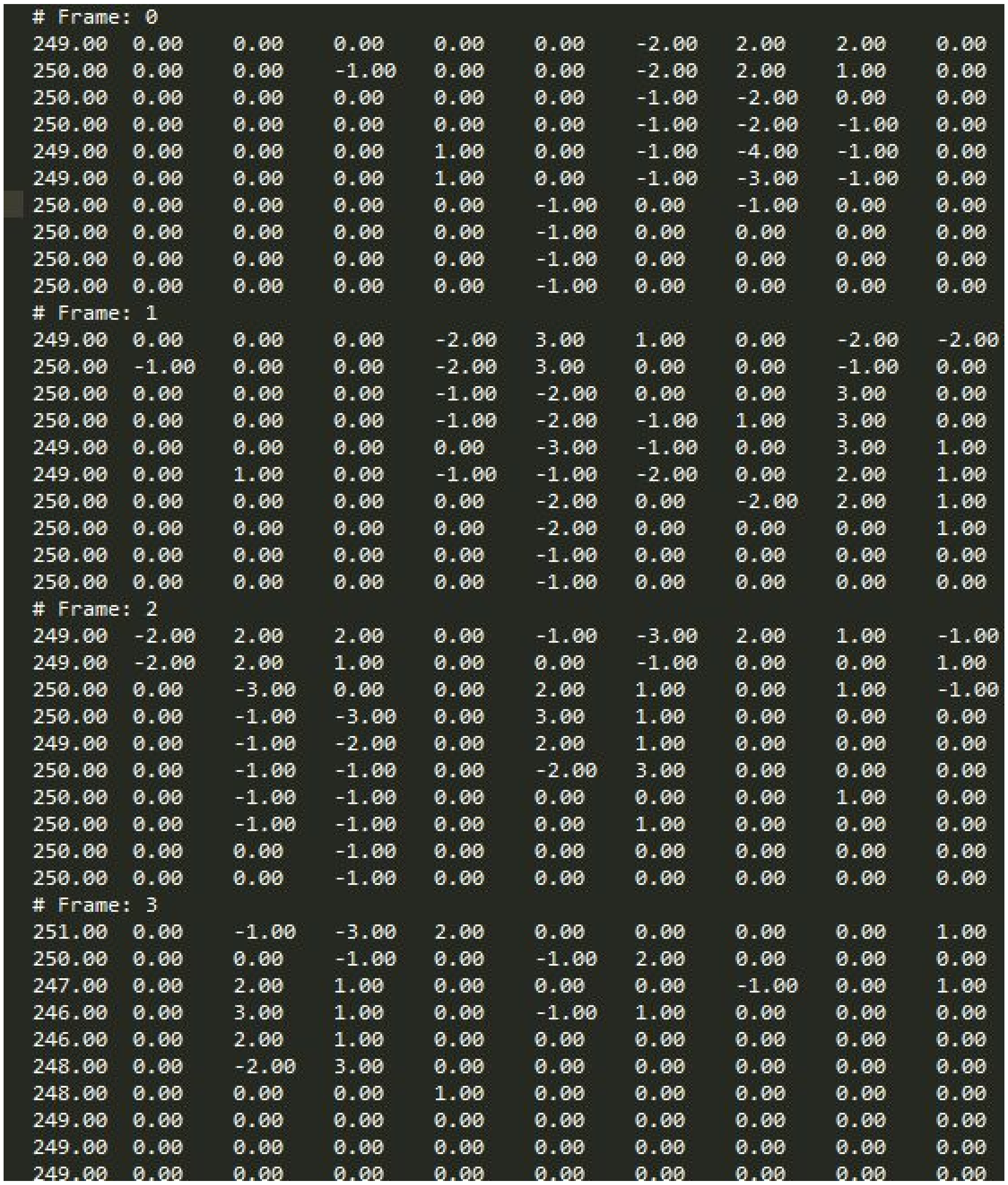
Running example is shown as follows.



```
C:\Anaconda\python.exe D:/YuhanSun/598/Phase2/Task2.py
### Spatial Predictive Coding [SPC] ###

Enter the make of the video file: 2.mp4

Enter the top-left X(Width) coordinate: 3
Enter the top-left Y(Height) coordinate: 10

### Reading the file ###

Select any one of the following:
Press 1 for No PC
Press 2 for Predictor A
Press 3 for Predictor B
Press 4 for Predictor C
Press 5 for Alpha-based Predictor
Choice : 1

### You select option 1 ###

### Calculating the encoding ###

### File saved as 2_1.spc ###

Prediction error is 0

Do you want to continue Y/N: Y
Enter the make of the video file: 2.mp4

Enter the top-left X(Width) coordinate: 3
Enter the top-left Y(Height) coordinate: 10

### Reading the file ###

Select any one of the following:
Press 1 for No PC
Press 2 for Predictor A
Press 3 for Predictor B
Press 4 for Predictor C
Press 5 for Alpha-based Predictor
Choice : 2

### You select option 2 ###

### Calculating the encoding ###

### File saved as 2_2.spc ###

Prediction error is 68908.0

Do you want to continue Y/N:
```



```
# Frame: 0
249.00  249.00  249.00  249.00  249.00  249.00  247.00  249.00  251.00  251.00
250.00  250.00  250.00  249.00  249.00  249.00  247.00  249.00  250.00  250.00
250.00  250.00  250.00  250.00  250.00  250.00  249.00  247.00  247.00  247.00
250.00  250.00  250.00  250.00  250.00  250.00  249.00  247.00  246.00  246.00
249.00  249.00  249.00  249.00  250.00  250.00  249.00  245.00  244.00  244.00
249.00  249.00  249.00  249.00  250.00  250.00  249.00  246.00  245.00  245.00
250.00  250.00  250.00  250.00  250.00  249.00  249.00  248.00  248.00  248.00
250.00  250.00  250.00  250.00  250.00  249.00  249.00  249.00  249.00  249.00
250.00  250.00  250.00  250.00  250.00  249.00  249.00  249.00  249.00  249.00
250.00  250.00  250.00  250.00  250.00  249.00  249.00  249.00  249.00  249.00
# Frame: 1
249.00  249.00  249.00  249.00  247.00  250.00  251.00  251.00  249.00  247.00
250.00  249.00  249.00  249.00  247.00  250.00  250.00  250.00  249.00  249.00
250.00  250.00  250.00  250.00  249.00  247.00  247.00  247.00  250.00  250.00
250.00  250.00  250.00  250.00  249.00  247.00  246.00  247.00  250.00  250.00
249.00  249.00  249.00  249.00  249.00  246.00  245.00  245.00  248.00  249.00
249.00  249.00  250.00  250.00  249.00  248.00  246.00  246.00  248.00  249.00
250.00  250.00  250.00  250.00  250.00  248.00  248.00  246.00  248.00  249.00
250.00  250.00  250.00  250.00  250.00  248.00  248.00  248.00  248.00  249.00
250.00  250.00  250.00  250.00  250.00  249.00  249.00  249.00  249.00  249.00
250.00  250.00  250.00  250.00  250.00  249.00  249.00  249.00  249.00  249.00
# Frame: 2
249.00  247.00  249.00  251.00  251.00  250.00  247.00  249.00  250.00  249.00
249.00  247.00  249.00  250.00  250.00  250.00  249.00  249.00  249.00  250.00
250.00  250.00  247.00  247.00  247.00  249.00  250.00  250.00  251.00  250.00
250.00  250.00  249.00  246.00  246.00  249.00  250.00  250.00  250.00  250.00
249.00  249.00  248.00  246.00  246.00  248.00  249.00  249.00  249.00  249.00
250.00  250.00  249.00  248.00  248.00  246.00  249.00  249.00  249.00  249.00
250.00  250.00  249.00  248.00  248.00  248.00  248.00  248.00  249.00  249.00
250.00  250.00  249.00  248.00  248.00  248.00  249.00  249.00  249.00  249.00
250.00  250.00  250.00  249.00  249.00  249.00  249.00  249.00  249.00  249.00
250.00  250.00  250.00  249.00  249.00  249.00  249.00  249.00  249.00  249.00
# Frame: 3
251.00  251.00  250.00  247.00  249.00  249.00  249.00  249.00  249.00  250.00
250.00  250.00  250.00  249.00  249.00  248.00  250.00  250.00  250.00  250.00
247.00  247.00  249.00  250.00  250.00  250.00  250.00  249.00  249.00  250.00
246.00  246.00  249.00  250.00  250.00  249.00  250.00  250.00  250.00  250.00
246.00  246.00  248.00  249.00  249.00  249.00  249.00  249.00  249.00  249.00
248.00  248.00  246.00  249.00  249.00  249.00  249.00  249.00  249.00  249.00
248.00  248.00  248.00  248.00  249.00  249.00  249.00  249.00  249.00  249.00
249.00  249.00  249.00  249.00  249.00  249.00  249.00  249.00  249.00  249.00
249.00  249.00  249.00  249.00  249.00  249.00  249.00  249.00  249.00  249.00
249.00  249.00  249.00  249.00  249.00  249.00  249.00  249.00  249.00  249.00
# Frame: 4
251.00  251.00  250.00  247.00  249.00  249.00  249.00  249.00  249.00  250.00
250.00  250.00  250.00  249.00  249.00  248.00  250.00  250.00  250.00  250.00
247.00  247.00  249.00  250.00  250.00  250.00  250.00  249.00  249.00  250.00
246.00  246.00  249.00  250.00  250.00  249.00  250.00  250.00  250.00  250.00
246.00  246.00  248.00  249.00  249.00  249.00  249.00  249.00  249.00  249.00
248.00  248.00  248.00  249.00  249.00  249.00  249.00  249.00  249.00  249.00
249.00  248.00  248.00  249.00  249.00  249.00  249.00  249.00  249.00  249.00
249.00  248.00  248.00  249.00  249.00  249.00  249.00  249.00  249.00  249.00
249.00  249.00  249.00  249.00  249.00  249.00  249.00  249.00  249.00  249.00
```



```
# Frame: 0
249.00  0.00    0.00    0.00    0.00    0.00    -2.00   2.00    2.00    0.00
250.00  0.00    0.00    -1.00   0.00    0.00    -2.00   2.00    1.00    0.00
250.00  0.00    0.00    0.00    0.00    0.00    -1.00   -2.00   0.00    0.00
250.00  0.00    0.00    0.00    0.00    0.00    -1.00   -2.00   -1.00   0.00
249.00  0.00    0.00    0.00    1.00    0.00    -1.00   -4.00   -1.00   0.00
249.00  0.00    0.00    0.00    1.00    0.00    -1.00   -3.00   -1.00   0.00
250.00  0.00    0.00    0.00    0.00    -1.00   0.00    -1.00   0.00    0.00
250.00  0.00    0.00    0.00    0.00    -1.00   0.00    0.00    0.00    0.00
250.00  0.00    0.00    0.00    0.00    -1.00   0.00    0.00    0.00    0.00
250.00  0.00    0.00    0.00    0.00    -1.00   0.00    0.00    0.00    0.00
# Frame: 1
249.00  0.00    0.00    0.00    -2.00   3.00    1.00    0.00    -2.00   -2.00
250.00  -1.00   0.00    0.00    -2.00   3.00    0.00    0.00    -1.00   0.00
250.00  0.00    0.00    0.00    -1.00   -2.00   0.00    0.00    3.00    0.00
250.00  0.00    0.00    0.00    -1.00   -2.00   -1.00   1.00    3.00    0.00
249.00  0.00    0.00    0.00    0.00    -3.00   -1.00   0.00    3.00    1.00
249.00  0.00    1.00    0.00    -1.00   -1.00   -2.00   0.00    2.00    1.00
250.00  0.00    0.00    0.00    0.00    -2.00   0.00    -2.00   2.00    1.00
250.00  0.00    0.00    0.00    0.00    -2.00   0.00    0.00    0.00    1.00
250.00  0.00    0.00    0.00    0.00    -1.00   0.00    0.00    0.00    0.00
250.00  0.00    0.00    0.00    0.00    -1.00   0.00    0.00    0.00    0.00
# Frame: 2
249.00  -2.00   2.00    2.00    0.00    -1.00   -3.00   2.00    1.00    -1.00
249.00  -2.00   2.00    1.00    0.00    0.00    -1.00   0.00    0.00    1.00
250.00  0.00    -3.00   0.00    0.00    2.00    1.00    0.00    1.00    -1.00
250.00  0.00    -1.00   -3.00   0.00    3.00    1.00    0.00    0.00    0.00
249.00  0.00    -1.00   -2.00   0.00    2.00    1.00    0.00    0.00    0.00
250.00  0.00    -1.00   -1.00   0.00    -2.00   3.00    0.00    0.00    0.00
250.00  0.00    -1.00   -1.00   0.00    0.00    0.00    0.00    1.00    0.00
250.00  0.00    -1.00   -1.00   0.00    0.00    1.00    0.00    0.00    0.00
250.00  0.00    0.00    -1.00   0.00    0.00    0.00    0.00    0.00    0.00
250.00  0.00    0.00    -1.00   0.00    0.00    0.00    0.00    0.00    0.00
# Frame: 3
251.00  0.00    -1.00   -3.00   2.00    0.00    0.00    0.00    0.00    1.00
250.00  0.00    0.00    -1.00   0.00    -1.00   2.00    0.00    0.00    0.00
247.00  0.00    2.00    1.00    0.00    0.00    0.00    -1.00   0.00    1.00
246.00  0.00    3.00    1.00    0.00    -1.00   1.00    0.00    0.00    0.00
246.00  0.00    2.00    1.00    0.00    0.00    0.00    0.00    0.00    0.00
248.00  0.00    -2.00   3.00    0.00    0.00    0.00    0.00    0.00    0.00
248.00  0.00    0.00    0.00    1.00    0.00    0.00    0.00    0.00    0.00
249.00  0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
249.00  0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
249.00  0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
```

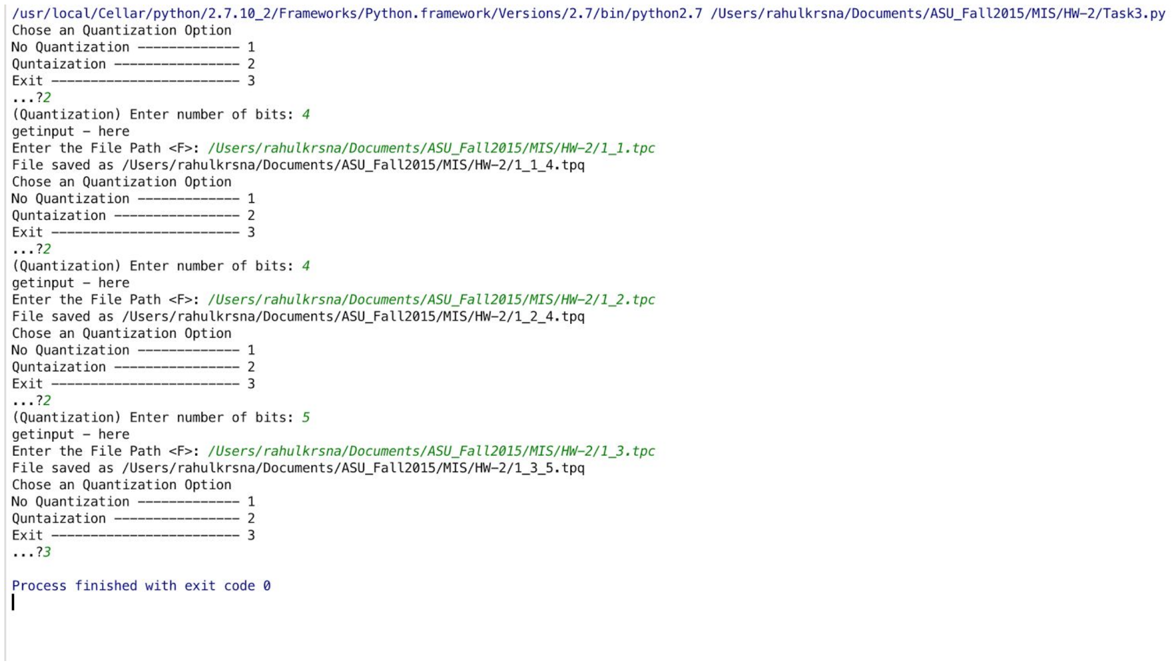
Task 3:
The objective of task 3 is to quantize the error values produced as output from Task-1 and Task-2

What?
Step 1: Input is taken from the user Input file and Quantization option and the number of bits
Step 2: Now a list of bins are created which will point to each location in the divided range of the possible error values (-255 to 255) equally. So for a particular error value a representative is given.

Step 3: Now we read the file chosen a input to the program, and quantize it based on the selected value for the range in which the value falls in.
Step 4: For all the error values the selected value replaces the actual error value to make the histogram more peakier, which is good for compression.
Step 5: The output is written back to the file in an X_Y_Z.tpq or X_Y_Z.spq where Z tells the number of bits chosen for quantization.

How?
1: For the step of finding if an error value falls in the specified quantized region, we initially create dictionary of all the possible values with the bucket number of the dictionary.
2: The error values are sorted to various bins using [np.digitize()] method and then the dictionary created is used to predict its selected value and then the actual value is replaced by the selected value
3: In case of number of bits more than 8, we quantize it to 1 bit quantization, because pow(2,9) is greater than the actual -255 to 255 range. (we are not considering float values in quantization ranges)

We use Task3_SPC.py file to compute quantization for the spacial encoding.


```
/usr/local/Cellar/python/2.7.10_2/Frameworks/Python.framework/Versions/2.7/bin/python2.7 /Users/rahulkrsna/Documents/ASU_Fall2015/MIS/HW-2/Task3.py
Chose an Quantization Option
No Quantization ------------- 1
Quntaization ---------------- 2
Exit ------------------------ 3
...?2
(Quantization) Enter number of bits: 4
getinput - here
Enter the File Path <F>: /Users/rahulkrsna/Documents/ASU_Fall2015/MIS/HW-2/1_1.tpc
File saved as /Users/rahulkrsna/Documents/ASU_Fall2015/MIS/HW-2/1_1_4.tpq
Chose an Quantization Option
No Quantization ------------- 1
Quntaization ---------------- 2
Exit ------------------------ 3
...?2
(Quantization) Enter number of bits: 4
getinput - here
Enter the File Path <F>: /Users/rahulkrsna/Documents/ASU_Fall2015/MIS/HW-2/1_2.tpc
File saved as /Users/rahulkrsna/Documents/ASU_Fall2015/MIS/HW-2/1_2_4.tpq
Chose an Quantization Option
No Quantization ------------- 1
Quntaization ---------------- 2
Exit ------------------------ 3
...?2
(Quantization) Enter number of bits: 5
getinput - here
Enter the File Path <F>: /Users/rahulkrsna/Documents/ASU_Fall2015/MIS/HW-2/1_3.tpc
File saved as /Users/rahulkrsna/Documents/ASU_Fall2015/MIS/HW-2/1_3_5.tpq
Chose an Quantization Option
No Quantization ------------- 1
Quntaization ---------------- 2
Exit ------------------------ 3
...?3

Process finished with exit code 0
```

Task 4:

Task 4 involves creating a bit stream according to choice of compression algorithm given the input of Task 3

Steps:

1. **No compression**

   Because there is no encoding, the input file is saved as it with the new file name. The input and output file sizes are the same.

   Output:

```
/***************************************************************/

input fileName: 2_1_0.tpq
option: 1
Size of input file in bytes:
184860
Size                    of                    output                    file
(/Users/kvivekanandan/Desktop/ASU/CSE_598_Multimedia_Information_Systems/Proje
ct/MIS_Phase_2/Submission/Code/2_1_0_1.tpv) in bytes:
184860

/***************************************************************/

/***************************************************************/

input fileName: 2_1_4.tpq
option: 1
Size of input file in bytes:
189303
Size                    of                    output                    file
(/Users/kvivekanandan/Desktop/ASU/CSE_598_Multimedia_Information_Systems/Proje
ct/MIS_Phase_2/Submission/Code/2_1_4_1.tpv) in bytes:
189303

/***************************************************************/

/***************************************************************/

input fileName: 2_1_0.spv
```

```
option: 1
Size of input file in bytes:
308830
Size of output file
(/Users/kvivekanandan/Desktop/ASU/CSE_598_Multimedia_Information_Systems/Proje
ct/MIS_Phase_2/Submission/Code/2_1_0_1.spv) in bytes:
308830

/**************************************************************/

/**************************************************************/

fileName: 2_1_4.spv
option: 1
Size of input file in bytes:
308830
Size of output file
(/Users/kvivekanandan/Desktop/ASU/CSE_598_Multimedia_Information_Systems/Proje
ct/MIS_Phase_2/Submission/Code/2_1_4_1.spv) in bytes:
308830

/**************************************************************/
```

2. Shannon-Fano

For a given list of symbols, develop a corresponding list of probabilities or frequency counts so that each symbol's relative frequency of occurrence is known.

1. Sort the lists of symbols according to frequency, with the most frequently occurring symbols at the left and the least common at the right.
2. Divide the list into two parts, with the total frequency counts of the left part being as close to the total of the right as possible.
3. The left part of the list is assigned the binary digit 0, and the right part is assigned the digit 1. This means that the codes for the symbols in the first part will all start with 0, and the codes in the second part will all start with 1.

4. Recursively apply the steps 3 and 4 to each of the two halves, subdividing groups and adding bits to the codes until each symbol has become a corresponding code leaf on the tree.

Output:

```
/**************************************************************/

input fileName: 2_1_0.tpq
option: 2
Size of input file in bytes:  184860
Size of compressed putput file
(/Users/kvivekanandan/Desktop/ASU/CSE_598_Multimedia_Information_Systems/Proje
ct/MIS_Phase_2/Submission/Code/2_1_0_2.tpv) in bytes:
86630

/**************************************************************/

/**************************************************************/

input fileName: 2_1_4.tpq
option: 2
Size of input file in bytes:  189303
Size of compressed putput file
(/Users/kvivekanandan/Desktop/ASU/CSE_598_Multimedia_Information_Systems/Proje
ct/MIS_Phase_2/Submission/Code/2_1_4_2.tpv) in bytes:
89504

/**************************************************************/

/**************************************************************/

input fileName: 2_1_0.spv
option: 2
Size of input file in bytes:  308830
Size of compressed putput file
(/Users/kvivekanandan/Desktop/ASU/CSE_598_Multimedia_Information_Systems/Proje
ct/MIS_Phase_2/Submission/Code/2_1_0_2.spv) in bytes:
162447

/**************************************************************/
/**************************************************************/
```

input fileName: **2_1_4.spv**
option: 2
Size of input file in bytes: 308830
Size of compressed putput file (/Users/kvivekanandan/Desktop/ASU/CSE_598_Multimedia_Information_Systems/Project/MIS_Phase_2/Submission/Code/2_1_4_2.spv) in bytes:
164570

/**/

References for implementation:

https://en.wikipedia.org/wiki/Shannon%E2%80%93Fano_coding

http://code.activestate.com/recipes/577502-shannon-fano-data-compression/

3. LZW:

A high level view of the encoding algorithm is shown here:

1. Initialize the dictionary to contain all strings of length one.
2. Find the longest string W in the dictionary that matches the current input.
3. Emit the dictionary index for W to output and remove W from the input.
4. Add W followed by the next symbol in the input to the dictionary.
5. Go to Step 2.

To decode an LZW-compressed archive, one needs to know in advance the initial dictionary used, but additional entries can be reconstructed as they are always simply concatenations of previous entries. At each stage, the decoder receives a code X; it looks X up in the table and outputs the sequence χ it codes, and it conjectures χ + ? as the entry the encoder just added – because the encoder emitted X for χ precisely because χ + ? was not in the table, and the encoder goes ahead and adds it.

The missing letter is the first letter in the sequence coded by the *next* code Z that the decoder receives. So the decoder looks up Z, decodes it into the sequence ω and takes the first letter z and tacks it onto the end of χ as the next dictionary entry.

When the decoder receives the code Z that is not yet in its dictionary:

1. The decoder sees X and then Z.
2. It knows X codes the sequence χ and Z codes some unknown sequence ω.

3. It knows the encoder just added Z to code χ + some unknown character,
4. and it knows that the unknown character is the first letter z of ω.
5. But the first letter of ω (= χ + ?) must then also be the first letter of χ.
6. So ω must be χ + x, where **x is the *first* letter of χ**.
7. So the decoder figures out what Z codes even though it's not in the table,
8. and upon receiving Z, the decoder decodes it as χ + x, and adds χ + x to the table as the value of Z.

Output

```
/*************************************************************/

input fileName: 2_1_0.tpq
option: 3
Size of input file in bytes:
184860
Size of compressed putput file
(/Users/kvivekanandan/Desktop/ASU/CSE_598_Multimedia_Information_Systems/Project/MI
S_Phase_2/Submission/Code/2_1_0_3.tpv) in bytes:
25085
/*************************************************************/


/*************************************************************/
input fileName: 2_1_4.tpq
option: 3
Size of input file in bytes:
189303
Size of compressed putput file
(/Users/kvivekanandan/Desktop/ASU/CSE_598_Multimedia_Information_Systems/Proje
ct/MIS_Phase_2/Submission/Code/2_1_4_3.tpv) in bytes:
7423

/*************************************************************/

/*************************************************************/

input fileName: 2_1_0.spv
option: 3
Size of input file in bytes:
308830
```

```
Size of compressed putput file (/Users/kvivekanandan/Desktop/ASU/CSE_598_Multimedia_Information_Systems/Project/MIS_Phase_2/Submission/Code/2_1_0_3.spv) in bytes:
29957

/**************************************************************/

/**************************************************************/

input fileName: 2_1_4.spv
option: 3
Size of input file in bytes:
308830
Size of compressed putput file (/Users/kvivekanandan/Desktop/ASU/CSE_598_Multimedia_Information_Systems/Project/MIS_Phase_2/Submission/Code/2_1_4_3.spv) in bytes:
13952

/**************************************************************/
```

References for implementation:

https://pypi.python.org/pypi/lzw/
https://en.wikipedia.org/wiki/Lempel%E2%80%93Ziv%E2%80%93Welch

4. Arithmetic encoding:

Arithmetic coding achieves compression by reducing the average number of bits required to represent a symbol. By assigning each symbol its own unique probability range, it's possible to encode a single symbol by its range. Using this approach, we encode a string as a series of probability ranges, but that doesn't compress anything. Instead additional symbols may be encoded by restricting the current probability range by the range of a new symbol being encoded. The pseudo code below illustrates how additional symbols may be added to an encoded string by restricting the string's range bounds.

```
lower bound = 0
upper bound = 1

while there are still symbols to encode
    current range = upper bound - lower bound
    upper bound = lower bound + (current range × upper bound of new symbol)
    lower bound = lower bound + (current range × lower bound of new symbol)
end while
```

Any value between the computed lower and upper probability bounds now encodes the input string.

The decoding process must start with a an encoded value representing a string. By definition, the encoded value lies within the lower and upper probability range bounds of the string it represents. Since the encoding process keeps restricting ranges (without shifting), the initial value also falls within the range of the first encoded symbol. Successive encoded symbols may be identified by removing the scaling applied by the known symbol. To do this, subtract out the lower probability range bound of the known symbol, and multiply by the size of the symbols' range.

Decoding a value may be performed following the steps in the pseudo code below:

```
encoded value = encoded input

while string is not fully decoded
    identify the symbol containing encoded value within its range

    //remove effects of symbol from encoded value
    current range = upper bound of new symbol - lower bound of new symbol
    encoded value = (encoded value - lower bound of new symbol) ÷ current range
end while
```

Output

```
/****************************************************************/
input fileName: 2_1_0.tpq
option: 4
Size of input file in bytes:
184860
Size of compressed putput file (/Users/kvivekanandan/Desktop/ASU/CSE_598_Multimedia_Information_Systems/Project/MIS_Phase_2/Submission/Code/2_1_0_4.tpv) in bytes:
71853
/****************************************************************/

/****************************************************************/
input fileName: 2_1_4.tpq
option: 4
Size of input file in bytes:
189303
```

```
Size of compressed putput file
(/Users/kvivekanandan/Desktop/ASU/CSE_598_Multimedia_Information_Systems/Proje
ct/MIS_Phase_2/Submission/Code/2_1_4_4.tpv) in bytes:
66468
/**************************************************************/
/**************************************************************/

input fileName: 2_1_0.spv
option: 4
Size of input file in bytes:
308830
Size of compressed putput file
(/Users/kvivekanandan/Desktop/ASU/CSE_598_Multimedia_Information_Systems/Proje
ct/MIS_Phase_2/Submission/Code/2_1_0_4.spv) in bytes:
108058

/**************************************************************/

/**************************************************************/

input fileName: 2_1_4.spv
option: 4
Size of input file in bytes:
308830
Size of compressed putput file
(/Users/kvivekanandan/Desktop/ASU/CSE_598_Multimedia_Information_Systems/Proje
ct/MIS_Phase_2/Submission/Code/2_1_4_4.spv) in bytes:
115331

/**************************************************************/
```

References for implementation:

https://pypi.python.org/pypi/arcode for Arithmetic encoding

Download and install bit manipulation library bitfile-0.2 from http://michael.dipperstein.com/bitlibs/ as dependency

Task 5:

Task 5 involves implementing a viewer that reads the binary file, and displays the decoded video.

Steps:

1. Decompress the output file of Task-4 to get encoded frames of the video file
    a. Parse the filename and call corresponding decompression method
    b. The decompression method outputs to a .tmp file (filename + "_dc)
2. Decode the encoded video file to get the quantized frames of the video file
    a. Calculate distortion, the Signal-to-Noise ration using the formula
        i. $$\mathrm{SNR} = \frac{P_{\mathrm{signal}}}{P_{\mathrm{noise}}} = \left(\frac{A_{\mathrm{signal}}}{A_{\mathrm{noise}}}\right)^2$$ SNR is the ratio of the average signal value μ_{sig} to the standard deviation of the *signal* σ_{sig}:
        ii. $$\mathrm{SNR} = \frac{\mu_{\mathrm{sig}}}{\sigma_{\mathrm{sig}}}$$ where SNR is the ratio of the average signal value μ_{sig} to the standard deviation of the *signal* σ_{sig}:
3. Stitch the frames together to create the output video file (.mp4)
    a. Reconstruct the signal from the quantized frames by using the corresponding predictive function formulas and output it to .tmp file
4. View displays the video
    a. Read the quantized frames and convert YUV to BGR and save it to video file
        i. Visualise as each frame is processed
            1. using cv2.imshow
        ii. Save it to output video file (output_*.mp4)
            1. cv2.VideoWriter
            2. using CV_FOURCC(*'**mp4v**') as the output format with frame rate as 20.0

Implementation Dependencies:

http://www.scipy.org/install.html

http://scikit-learn.org/stable/install.html

Program Execution:

All the programs are python execution files So python interpreter is required.

Sample images have been places as the output at the end of each task in the documentation for more reference

For the Program 1,2: A Video file needs to be provided as input.
For the Program 3: Input is output of Program1,2.
For the Program 4: Input is output of Program 3.
For the Program 5: Input is output of Program 4.

<u>Specific roles of the group members</u>

Group Member	**Role(s) Performed**	**Description**
Karen Tamayo	None	DROPPED THE COURSE
Marcelo Tejo	None	No deliverables
Karthik Vivekanandan	Documentation, Design, Development, Testing	Participated in the design discussion of all tasks, implementation of task 4 and 5 (flow, decompress, visualize), documentation for 4 and 5
Yuhan Sun	Design, Development, Code Review, Testing	Participated in the design discussion of all tasks, implementation of task 2 and 5 (decoding), documentation for 2
Rahul Krishna Vasantham	Documentation, Design, Development, Testing	Participated in the design discussion of all tasks, implementation of task 1, 3 and decoding in task 5, documentation for 1 and 3