PARALLEL AND DISTRIBUTED COMPUTING

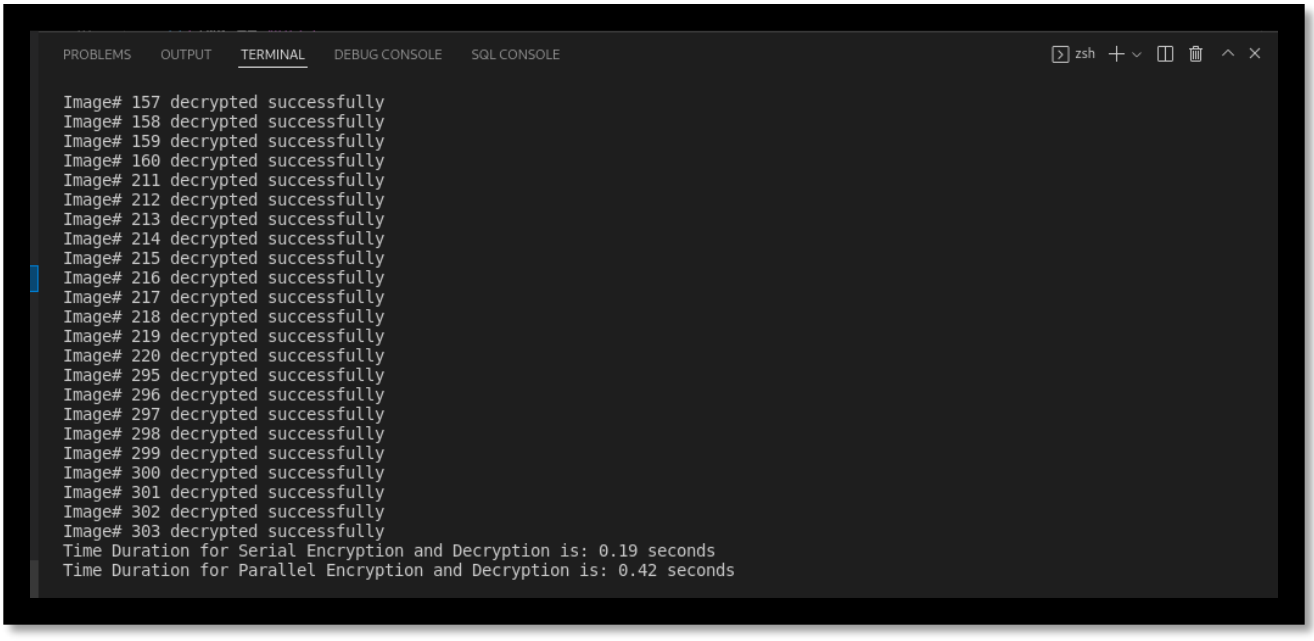
PROJECT SCREENSHOTS

Newton interpolation (Forward, Backward and Central) & Image Encryption and Decryption Algorithm

K190185
K190175
K190318
Section-G

Input: *having 2 data sets of 72mb and 456mb

Processing Input:



Parallel Code:

*using openmp

```
132  void encryptAllImagesParallel()
133  {
134      int n = fileCount("images");
135
136      printf("\nStarting Parallel Encryption\n\n");
137
138      omp_set_dynamic(0);
139      #pragma omp parallel for num_threads(64)
140      for(int i=1; i<=n; i++)
141      {
142          encryptImage(i);
143      }
144  }
145
146  void decryptAllImagesParallel()
147  {
148      int n = fileCount("encryptedImages");
149
150      printf("\nStarting Parallel Decryption\n\n");
151
152      omp_set_dynamic(0);
153      #pragma omp parallel for num_threads(64)
154      for(int i=1; i<=n; i++)
155      {
156          decryptImage(i);
157      }
158  }
```

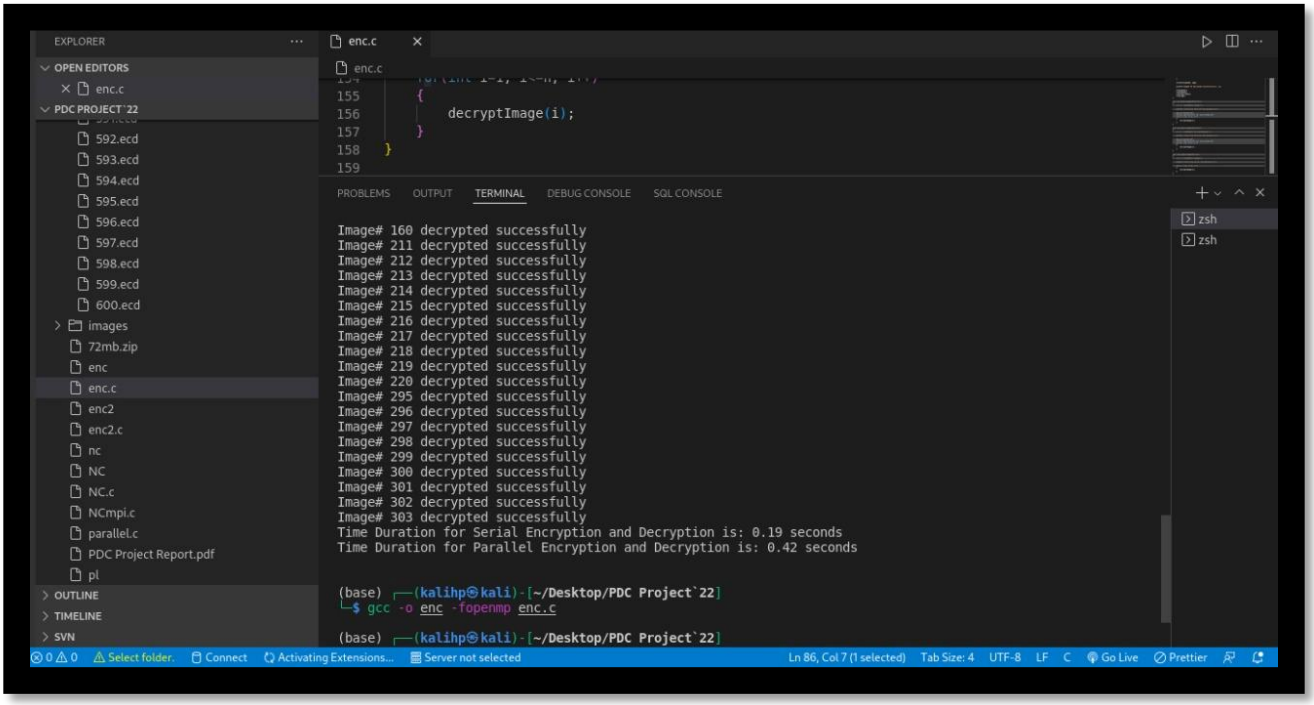
*Using MPI

```
51      //Intialize dataX elements with num1
52      int psize, my_rank, root_rank = 0, my_value;
53      MPI_Init(&argc, &argv);
54      MPI_Comm_size(MPI_COMM_WORLD, &psize);
55      if(mpisize != size)
56      {
57          printf("This application is meant to be run with size processes.\n");
58          MPI_Abort(MPI_COMM_WORLD, EXIT_FAILURE);
59      }
60      MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
61      my_value = (my_rank + 1) * 15;
62      if(my_rank == root_rank)
63      {
64          MPI_Gather(&my_value, 1, MPI_INT, dataY[][0], 1, MPI_INT, root_rank, MPI_COMM_WORLD);
65      }
66      else
67      {
68          MPI_Gather(&my_value, 1, MPI_INT, NULL, 0, MPI_INT, root_rank, MPI_COMM_WORLD);
69      }
70      MPI_Finalize();
71      //Intialize dataY Column 1 elements
72      MPI_Init(&argc, &argv);
73      MPI_Comm_size(MPI_COMM_WORLD, &psize);
74      if(mpisize != size)
75      {
76          printf("This application is meant to be run with size processes.\n");
77          MPI_Abort(MPI_COMM_WORLD, EXIT_FAILURE);
78      }
79      MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
80      my_value = (double)(rand() % 100) / 100000;
```

*Using serial code

```
159
160  void encryptAllImagesSerial()
161  {
162      int n = fileCount("images");
163
164      printf("\nStarting Serial Encryption\n\n");
165
166      for(int i=1; i<=n; i++)
167      {
168          encryptImage(i);
169      }
170  }
171
172  void decryptAllImagesSerial()
173  {
174      int n = fileCount("encryptedImages");
175
176      printf("\nStarting Serial Decryption\n\n");
177
178      for(int i=1; i<=n; i++)
179      {
180          decryptImage(i);
181      }
182  }
183
```

VS Code setup:



Compilation:

```
(base) ┌──(kalihp㉿kali)-[~/Desktop/PDC Project`22]
└─$ gcc -o enc -fopenmp enc.c

(base) ┌──(kalihp㉿kali)-[~/Desktop/PDC Project`22]
└─$ ./enc
```

Results: