

Name : Rejey Ezekiel

StudentID: 2348935

Class: DAAA/FT/2A/01

Assignment : ST1507_ASSIGNMENT_ONE_AY2425_SEM1

How to Operate my App:

1. Run python main.py

```
PS C:\Users\Rejey Ezekiel\Desktop\github\DSAA_CA1> python main.py

*****
* ST1507 DSAA MorseCode Message Analyzer *
* -----*
* *
* - Done by: Rejey Ezekiel Jeyakumar (2348935) *
* - Class : DAAA/FT/2A/01 *
*****
Press Enter to continue....
█
```

2. Press enter and select and option (Option: 1)

```
*****
* ST1507 DSAA MorseCode Message Analyzer *
* -----*
* *
* - Done by: Rejey Ezekiel Jeyakumar (2348935) *
* - Class : DAAA/FT/2A/01 *
*****
Press Enter to continue....

Please select your choice ('1','2','3','4','5','6','7'):
    1. Convert Text to Morse Code
    2. Convert Morse Code to Text
    3. Generate Morse Word Frequencies Report
    4. Generate Morse Keyword Frequencies Graph
    5. Morse to Text Character Analyzer
    6. Text to Morse Character Analyzer
    7. EXIT
Enter Choice: 1

Please Enter input file: █
```

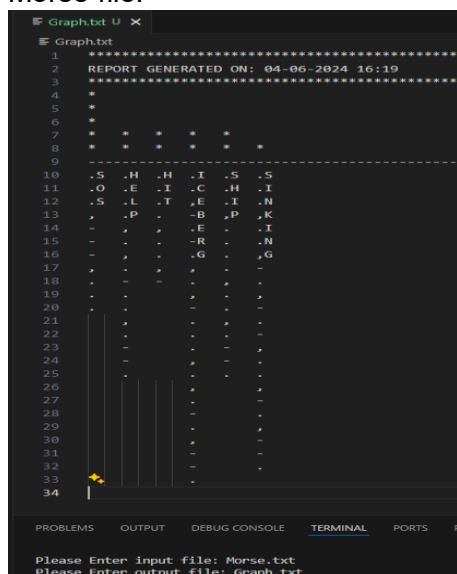
- a. Enter a Text file and a Output File :


```

Text.txt U  Report.txt M X
Report.txt
1 *****
2 REPORT GENERATED ON: 04-06-2024 16:17
3 *****
4
5 *** Decoded Morse Text
6 HELP US SOS SOS SOS
7 OUR SHIP HAS HIT AN ICEBERG
8 PLEASE HELP US
9 THIS IS A SOS
10 OUR SHIP IS SINKING
11 WE HIT AN ICEBERG
12 THIS IS AN SOS
13
14 *** Morse Words with frequency => 5
15 [...-.-.-.-.]=> SOS(*)
16
17 *** Morse Words with frequency => 3
18 [-.-.-.]=> AN
19 [...-.-.]=> IS
20
21 *** Morse Words with frequency => 2
22 [...-.-.-.-.-.]=> HELP(*)
23 [...-.-.-.-.]=> HIT(*)
24 [...-.-.-.-.-.-.-.-.-.-.]=> ICEBERG(*)
25 [-.-.-.-.-.]=> OUR
26 [...-.-.-.-.-.-.-.-.-.-.]=> SHIP(*)
27 [-.-.-.-.-.-.-.-.-.-.]=> THIS
28 [-.-.-.-.-.]=> US
29
30 *** Morse Words with frequency => 1
31 [-.-.]=> A
32 [...-.-.-.-.-.]=> HAS
33 [-.-.-.-.-.-.-.-.-.-.-.]=> PLEASE
34 [...-.-.-.-.-.-.-.-.-.-.-.]=> SINKING(*)
35 [-.-.-.-.-.]=> WE
36
37 *** Keywords sorted by frequency
38 SOS(5)
39 HELP(2)
40 HIT(2)
41 ICEBERG(2)
42 SHIP(2)
43 SINKING(1)
44
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POS
Please Enter input file: Morse.txt
Please Enter output file: Text.txt
Please select your choice ('1','2','3','4','5','6','7'):
1. Convert Text to Morse Code
2. Convert Morse Code to Text
3. Generate Morse Word Frequencies Report
4. Generate Morse Keyword Frequencies Graph
5. Morse to Text Character Analyzer
6. Text to Morse Character Analyzer
7. EXIT
Enter Choice: 3
Please Enter input file: Morse.txt
Please Enter output file: Report.txt

```

5. Select Option 4 to Generate Morse Keyword Frequencies Graph and input created Morse file.



6. Select Option 5 to view Text Character Analyzer

```
Please select your choice ('1','2','3','4','5','6','7'):  
1. Convert Text to Morse Code  
2. Convert Morse Code to Text  
3. Generate Morse Word Frequencies Report  
4. Generate Morse Keyword Frequencies Graph  
5. Morse to Text Character Analyzer  
6. Text to Morse Character Analyzer  
7. EXIT  
Enter Choice: 5  
  
Please Enter input file: Morse.txt  
  
Character Frequency Analysis:  
Character 'S': ***** (22)  
Character 'I': ***** (13)  
Character 'E': ***** (9)  
Character 'H': ***** (9)  
Character 'O': ***** (7)  
Character 'A': ***** (6)  
Character 'N': ***** (5)  
Character 'P': ***** (5)  
Character 'R': ***** (4)  
Character 'T': ***** (4)  
Character 'U': ***** (4)  
Character 'G': *** (3)  
Character 'L': *** (3)  
Character 'B': ** (2)  
Character 'C': ** (2)  
Character 'K': * (1)  
Character 'W': * (1)
```

7. Select Option 6 to view Morse Character Analyzer

```
Please select your choice ('1','2','3','4','5','6','7'):  
1. Convert Text to Morse Code  
2. Convert Morse Code to Text  
3. Generate Morse Word Frequencies Report  
4. Generate Morse Keyword Frequencies Graph  
5. Morse to Text Character Analyzer  
6. Text to Morse Character Analyzer  
7. EXIT  
Enter Choice: 6  
  
Please Enter input file: Text.txt  
  
Character Frequency Analysis:  
Character '... (S)': ***** (22)  
Character '.. (I)': ***** (13)  
Character '. (E)': ***** (9)  
Character '.... (H)': ***** (9)  
Character '--- (O)': ***** (7)  
Character '.- (A)': ***** (6)  
Character '-. (N)': ***** (5)  
Character '... (P)': ***** (5)  
Character '- (T)': ***** (4)  
Character '.-. (R)': ***** (4)  
Character '..- (U)': ***** (4)  
Character '--. (G)': ***** (3)  
Character '... (L)': ***** (3)  
Character '-.. (C)': ***** (2)  
Character '-... (B)': ***** (2)  
Character '-.- (K)': * (1)  
Character '--- (W)': * (1)
```

8. Lastly to exit the app press 7

```

Please select your choice ('1','2','3','4','5','6','7'):
    1. Convert Text to Morse Code
    2. Convert Morse Code to Text
    3. Generate Morse Word Frequencies Report
    4. Generate Morse Keyword Frequencies Graph
    5. Morse to Text Character Analyzer
    6. Text to Morse Character Analyzer
    7. EXIT
Enter Choice: 7
Bye, thanks for using ST1057 DSAA: MorseCode Message Analyzer
PS C:\Users\Rejey Ezekiel\Desktop\github\DSAA_CA1>

```

I have used many types of Inheritance like below :

```

"""New class Morse"""
class Morse:
    def __init__(self): ...
    """Function to initialize the Input File"""
    def select_input_file(self): ...

    """Function to initialize the Output File"""
    def select_output_file(self): ...

    """Function to convert a string from text to morse"""
    def morse_string(self,string): ...

    """Function to recursively call itself when the user enters a invalid file type"""
    def morse_recursive(self,string = None): ...

"""SubClasses of Morse"""
class Encoder(Morse):
    """Encoder Functionion to Encode Text into Morse"""
    def process(self):
        if '-' in self.file_contents:
            print('Invalid File Type')
            return self.morse_recursive()
        self.file_contents = self.file_contents.split('\n')
        morse_lines = []
        for line in self.file_contents:
            morse_line = ','.join(self.Morse_dict[char.upper()] for char in line if char.upper() in self.Morse_dict)
            morse_lines.append(morse_line)
        self.word = '\n'.join(morse_lines)

class Decoder(Morse):
    """Decoder Functionion to decode Morse to Text"""
    def process(self):
        self.file_contents = self.file_contents.replace('\n', ' , , ,').split(',')
        for i in self.file_contents:
            try:
                self.word += list(self.Morse_dict.keys())[list(self.Morse_dict.values()).index(i)]
            except:
                pass

```

I have also used Algorithms like Recursion to call itself and OOP approaches like Polymorphism:

```

"""Function to recursively call itself when the user enters a invalid file type"""
def morse_recursive(self,string = None):
    self.select_input_file()
    if Utility.CheckFileType(self.Input_File):
        self.file_contents = Utility.OpenTextFile(self.Input_File)
        self.process()
    else:
        self.morse_recursive()
    if string == None:
        state = True
        if state:
            self.select_output_file()
            if Utility.CheckFileType(self.Output_File,1):
                Utility.WriteFile(self.Output_File, self.word)
                state = False
        else:
            return self.word

```

Operation overloading for my word class:

```
class Word(Node):
    def __init__(self, name, frequency=1):
        self.original_name = name # Store original case
        self.name = name.lower().strip() # Normalize to lower case for comparison
        self.frequency = frequency
        super().__init__()

    def size(self):
        return len(self.name)

    def __eq__(self, otherNode):
        if otherNode is None:
            return False
        return self.name == otherNode.name

    def __lt__(self, otherNode):
        if otherNode is None:
            raise TypeError(
                "'<' not supported between instances of 'Word' and 'NoneType'"
            )
        if self.frequency == otherNode.frequency:
            return self.name < otherNode.name
        return self.frequency > otherNode.frequency

    def __str__(self):
        return f'{self.original_name}: {self.frequency}'
```

Private Function :

```
def __appendToHead(self, newNode):
    oldHeadNode = self.headNode
    self.headNode = newNode
    self.headNode.nextNode = oldHeadNode
    """
```

Initializing class with its super class :

```
class Report(Frequencies):

    """Initializes the Report class and its attributes."""
    def init(self):
        super().__init__()
        self.x = []
        self.frequency = None
        self.Default = None
```

For this assignment I have used 2 main Algorithms which are SortedList and Recursion.

Challenges I faced :

I felt that I didn't have a lot of time to implement more extravagant things as I had many issues where code wouldn't work here and would work at some other point therefore I had to rewrite my code many times to also follow the OOP standards.

Hierarchy of Classes :

