**Programming & Algorithms 2: Course Work 1 Report**

# 17 Enigma Machine Illustrations &amp; Clip Art - iStock

# Enigma Machine Encryption

Student Name: Ahmed Farouk Mahmoud

Student ID: CU2000512

Contents

[1](#_Toc87474163)

[Enigma Machine Encryption 1](#_Toc87474164)

[Introduction 3](#_Toc87474165)

[Expectations 3](#_Toc87474166)

[How to use the program 3](#_Toc87474167)

[Table of unite test 4](#_Toc87474168)

[Program explanation 4](#_Toc87474169)

[Dynamic array explanation 5](#_Toc87474170)

[classes and functions 5](#_Toc87474171)

[Objects and user inputs 5](#_Toc87474172)

[Starting the program 5](#_Toc87474173)

[Fixed array explanation 6](#_Toc87474174)

[Classes 6](#_Toc87474175)

[Objects 6](#_Toc87474176)

[Data Type – Rotor – 6](#_Toc87474177)

[Data structures and efficiency 6](#_Toc87474178)

[Data structures 6](#_Toc87474179)

[Efficiency 8](#_Toc87474180)

[Plug board configuration file 9](#_Toc87474181)

[Code for fixed array 9](#_Toc87474182)

[Code for dynamic array 23](#_Toc87474183)

# Introduction

In this course work we were asked to write code that modeled the enigma. The enigma was a cryptography machine used by the Germans in World War II to encrypt their messages although it was cracked after a few years of the war this machine shows the power of cryptography and its importance.

# Expectations

Before starting this course, I had a few expectations. Before jumping directly to the expectations, I need to explain where my level was before starting the course. I have built a grading system with python for last year’s project. I did not know anything about data structures, efficiency, or cryptography. I expected to learn and implement different data structures, understand efficiency and what makes one’s program more efficient than other programs. And understand some basic concepts about cryptography.

# How to use the program

The program is very user friendly to start the program run the python file

1. After running the code, you the user must choose one of three options for the plugboard

(1, 2, or 3) 1 will continue the program without a plugboard, 2 will use a predefined plugboard and 3 will allow you to configure your own plugboard.

* Note: if you choose 3 you must pare any 2 letters together in one input for example <ab> <BU> this means that (a) will be wired to (b) and (capital b) will be wired to (capital u)

1. After you configure the plugboard you have the option to choose 3 of 5 rotors to choose the rotors you can pic any 3 from (I, II, III, IV, V)

* Note: you only pic one rotor at once. For example IV, followed by an enter then II followed by an enter then V followed by enter. The above example means that I used (IV, II, V) IV as the first rotor, II as the second rotor, and V as the third and last rotor.

1. After choosing the 3 rotor that you will be asked to set the 3 rotors to any value from 0 to 62

* Note: this corresponds to the actual enigma where you had every rotor set to a certain number.
* Like Number 2 after setting a number you must press enter. For example, pressing 1 then enter, 5 then enter, 0 then enter means that rotor one is stet to 1 rotor two is set to 5 and rotor three is set to 0

1. Finally, you can type in the message that you would like to encrypt / decrypt

* Note you must memorize the above settings to decrypt the message

# Table of unite test

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Plugboard setting** | **Choosing the rotors** | **Setting the rotors** | **Message before encryption** | **Encrypted message** |
| 1 | I  II  III | 0  0  0 | Enigma encryption | OwjbfY hckWaMFPsq |
| 1 | II  V  IV | 1  2  3 | Today the weather was very nice | VsFiK g4e C8axh3z 5Zw L0sw B7lh |
| 2 | V  I  III | 9  6  3 | Hello world 1234 | bPZav n3JPv qxYU |
| 2 | V  IV  III | 9  9  9 | aaaaaaaaaaaaaaaaa | 9IHWFdkSebwADIozS  Note: this is to show that every (a) was encrypted to a different value |
| 3  [Custom plugboard configuration](#_Plug_board_configuration) | I  II  V | 3  3  8 | Ahmed Mohamed Farouk Mahmoud | LSl6u jwBqBtU 2lhjR5 NXyadfm |

# Program explanation

For this project I created 2 separate codes. One code with dynamic arrays (lists, dictionary) and the other with fixed arrays using the library Ctypes. Creating 2 separate programs helped me understand why data structures matter and how it can [affect efficiency](#_Data_structures_and). You can see both the codes inside my git repository.

Git repo ( <https://github.com/ahmedfarou22/enigmaa/> )

## Dynamic array explanation

### classes and functions

* There is only one class rotor (explained [below](#_Data_Type_Rotor))

Note: the rotors take in a number and return a different number (not letters)

* The function \_\_plug\_board\_\_ takes in a letter and returns the corresponding letter based on the configuration set by the user
* The function \_\_man\_reflector\_\_ takes in a number and reflects it back to the rotors

Note: if the reflector reflects 5 to 10 this means that 10 must be reflected to 5. In order for the machine to both encrypt and decrypt

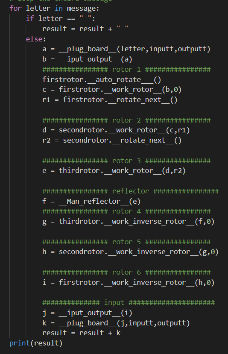
* Since the rotors only take numbers and the user inputs letter, I created the function \_\_input\_output\_\_ that takes the letter and translates it to a number and vice versa

### Objects and user inputs

* The rotor\_dictionary is a dictionary where I created all the 5 rotor objects and gave them keys
* The plug board section is the section where I take input form the user to configure the plugboard
* The next 3 parts are very similar I take input from the user for choosing and setting the rotors

### Starting the program

The following part is the easiest. The program loops around every letter in the message and runs the functions under in this fixed order

1. Plugboard function
2. Input function
3. First rotor
4. Second rotor
5. Third rotor
6. Reflector
7. Third rotor inverse
8. second rotor inverse
9. first rotor inverse
10. plugboard
11. output function

## Fixed array explanation

Like the dynamic array code this code is very similar except that it has 2 extra classes for the fixed arrays.

### Classes

* class array is a class that creates fixed arrays based on the length that the user specifies
* Class \_MapEntery is a class that takes a value and a key used for the dictionary

### Objects

* The above images show how I created all the fixed arrays needed in the program and how I stored all the map entries in a fixed array of 5

The rest of the code is the same as the dynamic code [explained above](#_Dynamic_array_explanation)

# Data Type – Rotor –

One of the challenges that I faced in this project was understanding how the machine in particular how-to rotors functioned. After a few failed attempt (you can see the failed attempt in a zip file named bad code .zip in my git repository) I came across a YouTube video that explained the enigma on a paper. I then printed this paper and started testing the paper enigma. This helped me visualize and understand how the machine – the rotors – worked. Which lead me to the next part. How to create my own data type from arrays that simulated a real-world rotor. It was apparent in the paper that every rotor had 2 repeating lists one ordered list and then another unordered list. This means that in the class Rotor it needed to take an input and an output list. I then needed to write a few functions to make the rotor work, rotate, and find the notch to send a signal to the next rotor to rotate. This part of the course work helped me understand and implement my own data type and understand why developers need different data types.

# Data structures and efficiency

## Data structures

Since the old programing languages only understood numbers and strings python comes with a few built-in data structures to help developers. For example, if a Farouk wants to create a program that keeps track of all the pets that he owned he can use the predefined data structure lists to achieve this task. So, a list is a data structure where multiple values can be stored together in a certain order.

Before starting the program, I had 3 options to choose from fixed array, dynamic array, or linked lists. The below figure explains some advantages and disadvantages of different Data structures.

|  |  |  |  |
| --- | --- | --- | --- |
| Data structure | Fixed Array | Dynamic Array | Linked lists |
| Advantages | Uses less memory | Has a dynamic size | Faster insertions, has a dynamic size |
| Disadvantages | Has a fixed size | Uses more memory | Uses the most memory. |

**Linked lists:**

linked lists allow for the fastest insertion O(1) since you just change the head without the need to loop around the entire array. It also has the advantage of using less memory because the program only takes the place that it needs. And gives the user the ability to change the size of the list.

**Dynamic Array:**

Dynamic arrays like lists and dictionaries in python, are grate options since it does not require a fixed size. However, this comes at a cost of memory usage since python takes a huge place in the memory for the dynamic array and when it needs more space it copies the entire memory slot.

**Fixed Array:**

Unlike dynamic arrays fixed arrays require a fixed size. And this allows for better memory usage since there will not be coping of the slot

**What I used for the enigma:**

Although at first you might think that linked lists are the best data structure to use for the enigma code since it uses the least memory, when you start to write the code you will realize that to rotate the rotors you need to save the .next of every item in the linked list. Which means that it will use even more memory. But have faster insertions. I then created 2 different codes one with fixed arrays and another with dynamic arrays.

## Efficiency

**Time complexity:**

To test the efficiency of the code I had to test 2 things time complexity and memory usage. The time complexity for both the programs is O(N). since I needed to loop around the 2 lists to get the encrypted value of the input. The only way to make this part faster was to sort the rotors then search with a binary search. However, this would not be useful in our cases since the entire project is built around the rotors being scrambled. And sorting them would just mean that (a) will be encrypted to (a) and (b) will be encrypted to (b) making the entire program useless.

|  |  |  |
| --- | --- | --- |
| Data Type | Fixed Array | Dynamic Array (lists, dictionary) |
| Full time to run the program (Message = Ahmed is the best one that says I love Egypt) | 0.0000992 seconds | 0.0000783 seconds |

As shown above the average time to run the 2 programs is almost the same both programs have a complicity of O(N)

**Memory usage:**

Although the time complexity would almost be the same in any data structure the memory usage is different. dynamic arrays would use the most memory followed by the linked lists and the most efficient would be the fixed array. See the figure below for the memory usage.

|  |  |  |
| --- | --- | --- |
| Data Type | Fixed Array | Dynamic Array (lists, dictionary) |
| Rotor out array | 120 bits | 568 bits |
| Rotor’s dictionary | 648 | 2,840 bits |

# Plug board configuration file

This is an example of option 3 in the plugboard you can copy and paste it into the program

an

bo

cp

dq

er

fs

gt

hu

iv

jw

kx

ly

mz

AN

BO

CP

DQ

ER

FS

GT

HU

IV

JW

KX

LY

MZ

09

18

27

36

45

# Code for fixed array

import ctypes

class \_MapEntry :

    def \_\_init\_\_(self, key, value):

        self.key = key

        self.value = value

    def \_\_str\_\_(self):

            return str(str(self.key)+" : "+ str(self.value))

class Array :

        def \_\_init\_\_( self, size ):

            assert size > 0, "Array size must be > 0"

            self.\_size = size

            PyArrayType = ctypes.py\_object \* size

            self.\_elements = PyArrayType()

            self.\_itemCount =0

        def \_\_len\_\_( self ):

            return self.\_size

        def \_\_print\_\_(self):

            for i in range(self.\_itemCount):

                print(self.\_elements[i], end =" ")

        def insertLast(self, item):

            if self.\_size > self.\_itemCount:

                self.\_elements[self.\_itemCount] = item

                self.\_itemCount +=1

            else:

                print("Array is full")

        def insertAll(self, item):

            for letter in item:

                if self.\_size > self.\_itemCount:

                    self.\_elements[self.\_itemCount] = letter

                    self.\_itemCount +=1

                else:

                    print("Array is full")

        def \_\_get\_value\_\_ (self,keyy):

            for item in self.\_elements:

                if keyy == item.key:

                    return item.value

        def linearSearch(self, term):

            for x in range(0,self.\_itemCount):

                if self.\_elements[x]== term:

                    return 1

            else:

                return 0

class rotor: # a data type that represents a rotor it takes in a number and outputs a number

    def \_\_init\_\_(self,rotor\_in,rotor\_out,rotor\_notch):

        self.rotor\_in =  rotor\_in.\_elements # a list

        self.rotor\_out = rotor\_out.\_elements # a list

        self.rotor\_notch = rotor\_notch # a letter

    # def \_\_inverse\_rotor\_\_(self):

    #      self.rotor\_notch = "#"

    #      return rotor(self.rotor\_out, self.rotor\_in, self.rotor\_notch)

    def \_\_auto\_rotate\_\_\_(self): # this function rotates the rotor once this is helpfull because after every letter is pressed the first rotor must rotate once

        self.rotor\_in = self.rotor\_in[1:] + self.rotor\_in[:1]      #rotate the rotor\_in once

        self.rotor\_out = self.rotor\_out[1:] + self.rotor\_out[:1]   #rotate the rotor\_out once

    def \_\_rotate\_next\_\_(self): # this function cheaks if the notch is at index zero if it is it returs 1 which is then used to rotate the next rotor before working it

        if self.rotor\_out[0] == self.rotor\_notch:

            return 1

        else:

            return 0

    def \_\_set\_rotor\_\_(self,setn): # this function is used to set the rotors before starting the machine it works by rotating the input and the output lists a certain number of unites

        self.rotor\_in = self.rotor\_in[setn:] + self.rotor\_in[:setn]

        self.rotor\_out = self.rotor\_out[setn:] + self.rotor\_out[:setn]

    def \_\_work\_rotor\_\_(self,number,rotatenext): # this is the main function it has 2 main parts

        if rotatenext == 1: # part one cheak if the rotor before sent 1 if it did then rotate the rotor once before execting the rest of the function

            self.rotor\_in = self.rotor\_in[1:] + self.rotor\_in[:1]

            self.rotor\_out = self.rotor\_out[1:] + self.rotor\_out[:1]

        value = 0  # part 2 of the function working the rotor

        value = self.rotor\_in[number]     # store the value of the letter that is iputed in a varuble named value

        for i in range(len(self.rotor\_out)): # loop around the rotor out and when the program finds the corsponding value return it

            if self.rotor\_out[i] == value:

                return i

    def \_\_work\_inverse\_rotor\_\_(self,number,rotatenext): # this is the main function it has 2 main parts

        if rotatenext == 1: # part one cheak if the rotor before sent 1 if it did then rotate the rotor once before execting the rest of the function

            self.rotor\_in = self.rotor\_in[1:] + self.rotor\_in[:1]

            self.rotor\_out = self.rotor\_out[1:] + self.rotor\_out[:1]

        value = 0  # part 2 of the function working the rotor

        value = self.rotor\_out[number]     # store the value of the letter that is iputed in a varuble named value

        for i in range(len(self.rotor\_in)): # loop around the rotor out and when the program finds the corsponding value return it

            if self.rotor\_in[i] == value:

                    return i

def \_\_plug\_board\_\_(letter,inputt,outputt):    # simple yet effective plug board

    # this plugboard works in a very simple way

    # it has 2 lists input and output

    # it cheaks the letter has what index in the input list and returns the value of the output list at the same index

    for i in range(len(inputt)):

        if letter == inputt.\_elements[i]:

            return outputt.\_elements[i]

def \_\_Man\_reflector\_\_(number): ## a reflector with hardcoded values

    # the easiest part of the program it takes a value and returns a diffrent one

    if number == 0:

        return 24

    if number == 1:

        return 17

    if number == 2:

        return 20

    if number == 3:

        return 7

    if number == 4:

        return 16

    if number == 5:

        return 18

    if number == 6:

        return 11

    if number == 7:

        return 3

    if number == 8:

        return 15

    if number == 9:

        return 23

    if number == 10:

        return 13

    if number == 11:

        return 6

    if number == 12:

        return 14

    if number == 13:

        return 10

    if number == 14:

        return 12

    if number == 15:

        return 8

    if number == 16:

        return 4

    if number == 17:

        return 1

    if number == 18:

        return 5

    if number == 19:

        return 25

    if number == 20:

        return 2

    if number == 21:

        return 22

    if number == 22:

        return 21

    if number == 23:

        return 9

    if number == 24:

        return 0

    if number == 25:

        return 19

    if number == 26:

        return 27

    if number == 27:

        return 26

    if number == 28:

        return 29

    if number == 29:

        return 28

    if number == 30:

        return 31

    if number == 31:

        return 30

    if number == 32:

        return 33

    if number == 33:

        return 32

    if number == 34:

        return 35

    if number == 35:

        return 36

    if number == 36:

        return 35

    if number == 37:

        return 38

    if number == 38:

        return 37

    if number == 39:

        return 40

    if number == 40:

        return 39

    if number == 41:

        return 42

    if number == 42:

        return 41

    if number == 43:

        return 44

    if number == 44:

        return 43

    if number == 45:

        return 46

    if number == 46:

        return 45

    if number == 47:

        return 48

    if number == 48:

        return 47

    if number == 49:

        return 50

    if number == 50:

        return 49

    if number == 51:

        return 52

    if number == 52:

        return 51

    if number == 53:

        return 54

    if number == 54:

        return 53

    if number == 55:

        return 56

    if number == 56:

        return 55

    if number == 57:

        return 58

    if number == 58:

        return 57

    if number == 59:

        return 60

    if number == 60:

        return 59

    if number == 61:

        return 61

def \_\_iput\_output\_\_(number\_or\_letter): # to make the program easier all the rotars take in a number and output a number this function turns letters to numbers and numbers to letters

    value = 0

    alphabet = rotor\_out

    if number\_or\_letter in alphabet.\_elements:

        #this is a letter lets convert to a number

        for i in range(len(alphabet.\_elements)):

            if alphabet.\_elements[i] == number\_or\_letter:

                value = i

                return i

    elif number\_or\_letter not in alphabet.\_elements:

        ## this is an intger lets convert it to a letter

         value = alphabet.\_elements[number\_or\_letter]

         return value

############## fixed arrays ######################

rotor\_out = Array(62)

rotor\_out.insertAll("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789")

rotor\_one\_in = Array(62)

rotor\_one\_in.insertAll("ekmflgdqvzntowyhxuspaibrcjBYCHGLFMNAOEPUXDWVKJIRSZTQ4573628910")

rotor\_two\_in = Array(62)

rotor\_two\_in.insertAll("ajdksiruxblhwtmcqgznpyfvoeCXJBOQPWAIMUVNHTEFZGSRKYLD0354216987")

rotor\_three\_in = Array(62)

rotor\_three\_in.insertAll("bdfhjlcprtxvznyeiwgakmusqoGHJFIBARUQTSPVWYOZXCNLMKDE2347695801")

rotor\_four\_in = Array(62)

rotor\_four\_in.insertAll("opqrxwvyutabzcdhsijklegmnfGAXFHIJEVWDTUBONQSCRLYZMPK1329876054")

rotor\_five\_in = Array(62)

rotor\_five\_in.insertAll("ikzwvujtsrqlmncdopefyxghbaZYXWVUTSRQPONMLKJIHGFEDCBA9876543210")

outputtt = Array(62)

outputtt.insertAll("nopqrstuvwxyzabcdefghijklmNOPQRSTUVWXYZABCDEFGHIJKLM9876543210")

################### 5 rotors ############### fixed

entry1 = \_MapEntry("I",  rotor(rotor\_one\_in,rotor\_out,"q"))

entry2 = \_MapEntry("II", rotor(rotor\_two\_in,rotor\_out,"e"))

entry3 = \_MapEntry("III",rotor(rotor\_three\_in,rotor\_out,"v"))

entry4 = \_MapEntry("IV", rotor(rotor\_four\_in,rotor\_out,"u"))

entry5 = \_MapEntry("V",  rotor(rotor\_five\_in,rotor\_out,"b"))

map1 = Array(5)

map1.insertLast(entry1)

map1.insertLast(entry2)

map1.insertLast(entry3)

map1.insertLast(entry4)

map1.insertLast(entry5)

# ==========================================================  program Starts Here =============================================================================

##################### configure the plug board ###############

print("plug board configuration")

while True:

    y\_n = str(input("you have 3 options press 1 continue witout a plugboard press 2 to continue with a predefined plugboard or press 3 to configure your own plugboard : "))

    if y\_n == "1":

        break

    elif y\_n == "2":

        break

    elif y\_n == "3":

        break

    else:

        print("please enter 1 2 or 3 : ")

        continue

if y\_n == "3":

    inputt = Array(62)

    outputt = Array(62)

    for i in range(0,31):

        while True:

            match = input("match 2 letters together Example <AB>: ")

            if len(match) > 2:

                print("please enter 2 values only")

                continue

            elif len(match) == 1:

                print(" please use two letters together")

                continue

            elif len(match) == 0:

                print(" please use two letters together")

                continue

            elif match[0] == match[1]:

                print(" do not be smart ha afshak")

                continue

            elif len(match) < 0:

                print(" soory i did not uderstand that")

                continue

            elif inputt.linearSearch(match[0]) == 1:

                print("you already assined this before")

            elif inputt.linearSearch(match[1]) == 1:

                print("you already assined this before")

            elif outputt.linearSearch(match[0]) == 1:

                print("you already assined this before")

            elif outputt.linearSearch(match[1]) == 1:

                print("you already assined this before")

            else:

                inputt.insertLast(match[0])

                outputt.insertLast(match[1])

                break

    for k in range(0,31):

        inputt.insertLast(outputt.\_elements[k])

        outputt.insertLast(inputt.\_elements[k])

if y\_n == "2":

    inputt  =  rotor\_out

    outputt = outputtt

elif y\_n == "1":

    print ("you did not configure a plug board we will proceed without a plug board")

    inputt  =  rotor\_out

    outputt = rotor\_out

###################### choose the 3 rotors ############### with validation on input

while True:

    choose\_the\_first\_rotor  = input("first position: choose a  rotor from the one of the five > I, II, III, IV, V : ").upper()

    if choose\_the\_first\_rotor == "I":

        break

    if choose\_the\_first\_rotor == "II":

        break

    if choose\_the\_first\_rotor == "III":

        break

    if choose\_the\_first\_rotor == "IV":

        break

    if choose\_the\_first\_rotor == "V":

        break

    else:

        print("Sorry I did not understand that please enter I, II, III, IV, or V")

while True:

    choose\_the\_second\_rotor = input("second position: choose a  rotor from the one of the five > I, II, III, IV, V : ").upper()

    if choose\_the\_second\_rotor == "I":

        break

    if choose\_the\_second\_rotor == "II":

        break

    if choose\_the\_second\_rotor == "III":

        break

    if choose\_the\_second\_rotor == "IV":

        break

    if choose\_the\_second\_rotor == "V":

        break

    else:

        print("Sorry I did not understand that please enter I, II, III, IV, or V")

while True:

    choose\_the\_third\_rotor  = input("third position: choose a  rotor from the one of the five > I, II, III, IV, V : ").upper()

    if choose\_the\_third\_rotor == "I":

        break

    if choose\_the\_third\_rotor == "II":

        break

    if choose\_the\_third\_rotor == "III":

        break

    if choose\_the\_third\_rotor == "IV":

        break

    if choose\_the\_third\_rotor == "V":

        break

    else:

        print("Sorry I did not understand that please enter I, II, III, IV, or V")

###################### set values to the 3 rotors ############### with validations

while True:

    try:

        set\_rotor\_one    = int(input("what number do you want to set rotor 1 to ? "))

    except ValueError:

        print("Sorry, I didn't understand that.")

        continue

    if set\_rotor\_one < 0:

        print("Sorry, your response must not be negative.")

        continue

    if set\_rotor\_one > 62:

        print("Sorry, your response is bigger than 62.")

        continue

    else:

        break

while True:

    try:

        set\_rotor\_two    = int(input("what number do you want to set rotor 2 to ? "))

    except ValueError:

        print("Sorry, I didn't understand that.")

        continue

    if set\_rotor\_two < 0:

        print("Sorry, your response must not be negative.")

        continue

    if set\_rotor\_two > 62:

        print("Sorry, your response is bigger than 62.")

        continue

    else:

        break

while True:

    try:

        set\_rotor\_three  = int(input("what number do you want to set rotor 3 to ? "))

    except ValueError:

        print("Sorry, I didn't understand that.")

        continue

    if set\_rotor\_three < 0:

        print("Sorry, your response must not be negative.")

        continue

    if set\_rotor\_three > 62:

        print("Sorry, your response is bigger than 62.")

        continue

    else:

        break

while True: # take the message and validate any spcial caracters

    message = str(input("what is the message you want to encript/decript "))

    if any(c in rotor\_out.\_elements for c in message):

        break

    else:

        print("Your input contains values that are not suported please try agin")

        continue

result = ""

###################### create all rotors and give them set values ###############

firstrotor = map1.\_\_get\_value\_\_(choose\_the\_first\_rotor)

firstrotor.\_\_set\_rotor\_\_(set\_rotor\_one)

secondrotor =map1.\_\_get\_value\_\_(choose\_the\_second\_rotor)

secondrotor.\_\_set\_rotor\_\_(set\_rotor\_two)

thirdrotor = map1.\_\_get\_value\_\_(choose\_the\_third\_rotor)

thirdrotor.\_\_set\_rotor\_\_(set\_rotor\_three)

# loop the entire message

for letter in message:

    if letter == " ":

        result = result + " "

    else:

        a = \_\_plug\_board\_\_(letter,inputt,outputt)

        b = \_\_iput\_output\_\_(a)

        ################ rotor 1 ################

        firstrotor.\_\_auto\_rotate\_\_\_()

        c = firstrotor.\_\_work\_rotor\_\_(b,0)

        r1 = firstrotor.\_\_rotate\_next\_\_()

        ################ rotor 2 ################

        d = secondrotor.\_\_work\_rotor\_\_(c,r1)

        r2 = secondrotor.\_\_rotate\_next\_\_()

        ################ rotor 3 ################

        e = thirdrotor.\_\_work\_rotor\_\_(d,r2)

        ################ reflector ################

        f = \_\_Man\_reflector\_\_(e)

        ################ rotor 4 ################

        g = thirdrotor.\_\_work\_inverse\_rotor\_\_(f,0)

        ################ rotor 5 ################

        h = secondrotor.\_\_work\_inverse\_rotor\_\_(g,0)

        ################ rotor 6 ################

        i = firstrotor.\_\_work\_inverse\_rotor\_\_(h,0)

        ############## input #####################

        j = \_\_iput\_output\_\_(i)

        k = \_\_plug\_board\_\_(j,inputt,outputt)

        result = result + k

print(result)

# to test memory use uncomment the line under

# import os, psutil; print(psutil.Process(os.getpid()).memory\_info().rss / 1024 \*\* 2)

# Code for dynamic array

import time

class rotor: # a data type that represents a rotor it takes in a number and outputs a number

    def \_\_init\_\_(self,rotor\_in,rotor\_out,rotor\_notch):

        self.rotor\_in =  rotor\_in # a list

        self.rotor\_out = rotor\_out # a list

        self.rotor\_notch = rotor\_notch # a letter

    # def \_\_inverse\_rotor\_\_(self):

    #      self.rotor\_notch = "#"

    #      return rotor(self.rotor\_out, self.rotor\_in, self.rotor\_notch)

    def \_\_auto\_rotate\_\_\_(self): # this function rotates the rotor once this is helpfull because after every letter is pressed the first rotor must rotate once

        self.rotor\_in = self.rotor\_in[1:] + self.rotor\_in[:1]      #rotate the rotor\_in once

        self.rotor\_out = self.rotor\_out[1:] + self.rotor\_out[:1]   #rotate the rotor\_out once

    def \_\_rotate\_next\_\_(self): # this function cheaks if the notch is at index zero if it is it returs 1 which is then used to rotate the next rotor before working it

        if self.rotor\_out[0] == self.rotor\_notch:

            return 1

        else:

            return 0

    def \_\_set\_rotor\_\_(self,setn): # this function is used to set the rotors before starting the machine it works by rotating the input and the output lists a certain number of unites

        self.rotor\_in = self.rotor\_in[setn:] + self.rotor\_in[:setn]

        self.rotor\_out = self.rotor\_out[setn:] + self.rotor\_out[:setn]

    def \_\_work\_rotor\_\_(self,number,rotatenext): # this is the main function it has 2 main parts

        if rotatenext == 1: # part one cheak if the rotor before sent 1 if it did then rotate the rotor once before execting the rest of the function

            self.rotor\_in = self.rotor\_in[1:] + self.rotor\_in[:1]

            self.rotor\_out = self.rotor\_out[1:] + self.rotor\_out[:1]

        value = 0  # part 2 of the function working the rotor

        value = self.rotor\_in[number]     # store the value of the letter that is iputed in a varuble named value

        for i in range(len(self.rotor\_out)): # loop around the rotor out and when the program finds the corsponding value return it

            if self.rotor\_out[i] == value:

                return i

    def \_\_work\_inverse\_rotor\_\_(self,number,rotatenext): # this is the main function it has 2 main parts

        if rotatenext == 1: # part one cheak if the rotor before sent 1 if it did then rotate the rotor once before execting the rest of the function

            self.rotor\_in = self.rotor\_in[1:] + self.rotor\_in[:1]

            self.rotor\_out = self.rotor\_out[1:] + self.rotor\_out[:1]

        value = 0  # part 2 of the function working the rotor

        value = self.rotor\_out[number]     # store the value of the letter that is iputed in a varuble named value

        for i in range(len(self.rotor\_in)): # loop around the rotor out and when the program finds the corsponding value return it

            if self.rotor\_in[i] == value:

                    return i

def \_\_plug\_board\_\_(letter,inputt,outputt):    # simple yet effective plug board

    # this plugboard works in a very simple way

    # it has 2 lists input and output

    # it cheaks the letter has what index in the input list and returns the value of the output list at the same index

    for i in range(len(inputt)):

        if letter == inputt[i]:

            return outputt[i]

def \_\_Man\_reflector\_\_(number): ## a reflector with hardcoded values

    # the easiest part of the program it takes a value and returns a diffrent one

    if number == 0:

        return 24

    if number == 1:

        return 17

    if number == 2:

        return 20

    if number == 3:

        return 7

    if number == 4:

        return 16

    if number == 5:

        return 18

    if number == 6:

        return 11

    if number == 7:

        return 3

    if number == 8:

        return 15

    if number == 9:

        return 23

    if number == 10:

        return 13

    if number == 11:

        return 6

    if number == 12:

        return 14

    if number == 13:

        return 10

    if number == 14:

        return 12

    if number == 15:

        return 8

    if number == 16:

        return 4

    if number == 17:

        return 1

    if number == 18:

        return 5

    if number == 19:

        return 25

    if number == 20:

        return 2

    if number == 21:

        return 22

    if number == 22:

        return 21

    if number == 23:

        return 9

    if number == 24:

        return 0

    if number == 25:

        return 19

    if number == 26:

        return 27

    if number == 27:

        return 26

    if number == 28:

        return 29

    if number == 29:

        return 28

    if number == 30:

        return 31

    if number == 31:

        return 30

    if number == 32:

        return 33

    if number == 33:

        return 32

    if number == 34:

        return 35

    if number == 35:

        return 36

    if number == 36:

        return 35

    if number == 37:

        return 38

    if number == 38:

        return 37

    if number == 39:

        return 40

    if number == 40:

        return 39

    if number == 41:

        return 42

    if number == 42:

        return 41

    if number == 43:

        return 44

    if number == 44:

        return 43

    if number == 45:

        return 46

    if number == 46:

        return 45

    if number == 47:

        return 48

    if number == 48:

        return 47

    if number == 49:

        return 50

    if number == 50:

        return 49

    if number == 51:

        return 52

    if number == 52:

        return 51

    if number == 53:

        return 54

    if number == 54:

        return 53

    if number == 55:

        return 56

    if number == 56:

        return 55

    if number == 57:

        return 58

    if number == 58:

        return 57

    if number == 59:

        return 60

    if number == 60:

        return 59

    if number == 61:

        return 61

def \_\_iput\_output\_\_(number\_or\_letter): # to make the program easier all the rotars take in a number and output a number this function turns letters to numbers and numbers to letters

    value = 0

    alphabet = ["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z","A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T","U","V","W","X","Y","Z","0","1","2","3","4","5","6","7","8","9"]

    if number\_or\_letter in alphabet:

        #this is a letter lets convert to a number

        for i in range(len(alphabet)):

            if alphabet[i] == number\_or\_letter:

                value = i

                return i

    elif number\_or\_letter not in alphabet:

        ## this is an intger lets convert it to a letter

         value = alphabet[number\_or\_letter]

         return value

################### 5 rotors ###############

rotor\_dictionary = {

  "I"   : rotor(["e","k","m","f","l","g","d","q","v","z","n","t","o","w","y","h","x","u","s","p","a","i","b","r","c","j","B","Y","C","H","G","L","F","M","N","A","O","E","P","U","X","D","W","V","K","J","I","R","S","Z","T","Q","4","5","7","3","6","2","8","9","1","0"],["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z","A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T","U","V","W","X","Y","Z","0","1","2","3","4","5","6","7","8","9"],"q"),

  "II"  : rotor(["a","j","d","k","s","i","r","u","x","b","l","h","w","t","m","c","q","g","z","n","p","y","f","v","o","e","C","X","J","B","O","Q","P","W","A","I","M","U","V","N","H","T","E","F","Z","G","S","R","K","Y","L","D","0","3","5","4","2","1","6","9","8","7"],["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z","A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T","U","V","W","X","Y","Z","0","1","2","3","4","5","6","7","8","9"],"e"),

  "III" : rotor(["b","d","f","h","j","l","c","p","r","t","x","v","z","n","y","e","i","w","g","a","k","m","u","s","q","o","G","H","J","F","I","B","A","R","U","Q","T","S","P","V","W","Y","O","Z","X","C","N","L","M","K","D","E","2","3","4","7","6","9","5","8","0","1"],["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z","A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T","U","V","W","X","Y","Z","0","1","2","3","4","5","6","7","8","9"],"v"),

  "IV"  : rotor(["o","p","q","r","x","w","v","y","u","t","a","b","z","c","d","h","s","i","j","k","l","e","g","m","n","f","G","A","X","F","H","I","J","E","V","W","D","T","U","B","O","N","Q","S","C","R","L","Y","Z","M","P","K","1","3","2","9","8","7","6","0","5","4"],["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z","A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T","U","V","W","X","Y","Z","0","1","2","3","4","5","6","7","8","9"],"u"),

  "V"   : rotor(["i","k","z","w","v","u","j","t","s","r","q","l","m","n","c","d","o","p","e","f","y","x","g","h","b","a","Z","Y","X","W","V","U","T","S","R","Q","P","O","N","M","L","K","J","I","H","G","F","E","D","C","B","A","9","8","7","6","5","4","3","2","1","0"],["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z","A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T","U","V","W","X","Y","Z","0","1","2","3","4","5","6","7","8","9"],"b")

}

# ==========================================================  program Starts Here =============================================================================

##################### configure the plug board ###############

print("plug board configuration")

while True:

    y\_n = str(input("you have 3 options press 1 continue witout a plugboard press 2 to continue with a predefined plugboard or press 3 to configure your own plugboard : "))

    if y\_n == "1":

        break

    elif y\_n == "2":

        break

    elif y\_n == "3":

        break

    else:

        print("please enter 1 2 or 3 : ")

        continue

if y\_n == "3":

    inputt = []

    outputt = []

    for i in range(0,31):

        while True:

            match = input("match 2 letters together Example <AB>: ")

            if len(match) > 2:

                print("please enter 2 values only")

                continue

            elif len(match) == 1:

                print(" please use two letters together")

                continue

            elif len(match) == 0:

                print(" please use two letters together")

                continue

            elif len(match) < 0:

                print(" soory i did not uderstand that")

                continue

            elif match[0] in inputt:

                print("you already assined this before")

            elif match[1] in inputt:

                print("you already assined this before")

            elif match[0] in outputt:

                print("you already assined this before")

            elif match[1] in outputt:

                print("you already assined this before")

            else:

                inputt.append(match[0])

                outputt.append(match[1])

                break

    for k in range(0,31):

        inputt.append(outputt[k])

        outputt.append(inputt[k])

if y\_n == "2":

    inputt  =  ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z','A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z',"0","1","2","3","4","5","6","7","8","9"]

    outputt =  ['n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm','N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M',"9","8","7","6","5","4","3","2","1","0"]

elif y\_n == "1":

    print ("you did not configure a plug board we will proceed without a plug board")

    inputt  =  ["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z","A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T","U","V","W","X","Y","Z","0","1","2","3","4","5","6","7","8","9"]

    outputt =  ["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z","A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T","U","V","W","X","Y","Z","0","1","2","3","4","5","6","7","8","9"]

###################### choose the 3 rotors ############### with validation on input

while True:

    choose\_the\_first\_rotor  = input("first position: choose a  rotor from the one of the five > I, II, III, IV, V : ").upper()

    if choose\_the\_first\_rotor == "I":

        break

    if choose\_the\_first\_rotor == "II":

        break

    if choose\_the\_first\_rotor == "III":

        break

    if choose\_the\_first\_rotor == "IV":

        break

    if choose\_the\_first\_rotor == "V":

        break

    else:

        print("Sorry I did not understand that please enter I, II, III, IV, or V")

while True:

    choose\_the\_second\_rotor = input("second position: choose a  rotor from the one of the five > I, II, III, IV, V : ").upper()

    if choose\_the\_second\_rotor == "I":

        break

    if choose\_the\_second\_rotor == "II":

        break

    if choose\_the\_second\_rotor == "III":

        break

    if choose\_the\_second\_rotor == "IV":

        break

    if choose\_the\_second\_rotor == "V":

        break

    else:

        print("Sorry I did not understand that please enter I, II, III, IV, or V")

while True:

    choose\_the\_third\_rotor  = input("third position: choose a  rotor from the one of the five > I, II, III, IV, V : ").upper()

    if choose\_the\_third\_rotor == "I":

        break

    if choose\_the\_third\_rotor == "II":

        break

    if choose\_the\_third\_rotor == "III":

        break

    if choose\_the\_third\_rotor == "IV":

        break

    if choose\_the\_third\_rotor == "V":

        break

    else:

        print("Sorry I did not understand that please enter I, II, III, IV, or V")

###################### set values to the 3 rotors ############### with validations

while True:

    try:

        set\_rotor\_one    = int(input("what number do you want to set rotor 1 to ? "))

    except ValueError:

        print("Sorry, I didn't understand that.")

        continue

    if set\_rotor\_one < 0:

        print("Sorry, your response must not be negative.")

        continue

    if set\_rotor\_one > 62:

        print("Sorry, your response is bigger than 62.")

        continue

    else:

        break

while True:

    try:

        set\_rotor\_two    = int(input("what number do you want to set rotor 2 to ? "))

    except ValueError:

        print("Sorry, I didn't understand that.")

        continue

    if set\_rotor\_two < 0:

        print("Sorry, your response must not be negative.")

        continue

    if set\_rotor\_two > 62:

        print("Sorry, your response is bigger than 62.")

        continue

    else:

        break

while True:

    try:

        set\_rotor\_three  = int(input("what number do you want to set rotor 3 to ? "))

    except ValueError:

        print("Sorry, I didn't understand that.")

        continue

    if set\_rotor\_three < 0:

        print("Sorry, your response must not be negative.")

        continue

    if set\_rotor\_three > 62:

        print("Sorry, your response is bigger than 62.")

        continue

    else:

        break

while True: # take the message and validate any spcial caracters

    message = str(input("what is the message you want to encript/decript "))

    if any(c in inputt for c in message):

        break

    else:

        print("Your input contains values that are not suported please try agin")

        continue

result = ""

###################### create all rotors and give them set values ###############

firstrotor = rotor\_dictionary[choose\_the\_first\_rotor]

firstrotor.\_\_set\_rotor\_\_(set\_rotor\_one)

secondrotor =rotor\_dictionary[choose\_the\_second\_rotor]

secondrotor.\_\_set\_rotor\_\_(set\_rotor\_two)

thirdrotor = rotor\_dictionary[choose\_the\_third\_rotor]

thirdrotor.\_\_set\_rotor\_\_(set\_rotor\_three)

# loop the entire message

for letter in message:

    if letter == " ":

        result = result + " "

    else:

        a = \_\_plug\_board\_\_(letter,inputt,outputt)

        b = \_\_iput\_output\_\_(a)

        ################ rotor 1 ################

        firstrotor.\_\_auto\_rotate\_\_\_()

        c = firstrotor.\_\_work\_rotor\_\_(b,0)

        r1 = firstrotor.\_\_rotate\_next\_\_()

        ################ rotor 2 ################

        d = secondrotor.\_\_work\_rotor\_\_(c,r1)

        r2 = secondrotor.\_\_rotate\_next\_\_()

        ################ rotor 3 ################

        e = thirdrotor.\_\_work\_rotor\_\_(d,r2)

        ################ reflector ################

        f = \_\_Man\_reflector\_\_(e)

        ################ rotor 4 ################

        g = thirdrotor.\_\_work\_inverse\_rotor\_\_(f,0)

        ################ rotor 5 ################

        h = secondrotor.\_\_work\_inverse\_rotor\_\_(g,0)

        ################ rotor 6 ################

        i = firstrotor.\_\_work\_inverse\_rotor\_\_(h,0)

        ############## input #####################

        j = \_\_iput\_output\_\_(i)

        k = \_\_plug\_board\_\_(j,inputt,outputt)

        result = result + k

print(result)

# to test memory use uncomment the line under

# import os, psutil; print(psutil.Process(os.getpid()).memory\_info().rss / 1024 \*\* 2)