# Hannah Carl

# CS 345

### HW 1

Part 1: This password cracker could be more efficient if I were to use multi-threading or an additional source like a rainbow table. This password cracker would also be much quicker, if there was some knowledge known about the passwords like a hint to limit the brute force options. I utilized counters and if statements to ensure the password cracker would end once all passwords have been solved or all combinations have been exhausted to keep the process as quick as possible and to prevent infinite checks. I also tried to implement additional rules into the word lists based on english grammar and use only one word from the dictionary at a time with all of the changes to make the process more efficient. Within each check, I try to compare the passwords I am given with the hash I produce from the given rule, in order to prevent the rules from needing to create the same combination more than once.

#### Part 2:

- 1) DES
  - a) (weak DES keys) There are four so called weak DES keys. One of those keys is K= 0001111100011111 00011111 00011111 00001110 00001110 00001110 00001110.
  - b) What happens if you use this key?
    - i) If you use this key, then the left circular shift operation in DES would not have any effect due to the fact that each half of the bits are identical.
    - ii) After the sixteen DES rounds, these weak keys would result in 16 identical subkeys, thus being a very weak key for decryption.
  - c) Can you find the other three weak keys?

### Part 3:

- 1) Encoding ASCII (5 pts)
  - a) Decoded to: Le flag de ce challenge est: 2ac376481ae546cd689d5b91275d324e

    Results Name of the event validations Number of points Difficulty DA Author
  - Results Name of the event validations Number of points @ Difficulty @ Author Note @ Solution

    b) Encoding ASCII 27% 30460 5 ... Xartrick 😂 14

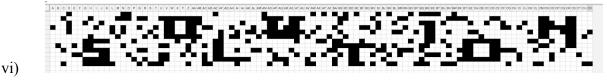
Do not forget to rate this challenge by giving y	your opinion ;-)	
Enter the password:		

- d) I used the associated resource (ASCII Table) to decode the message.
- e) I started on the left and slowly wrote down each letter or number based on the pair of two characters in the string and would highlight what I have already decoded.

### 2) Pixelated Decomposition (15pts)



- c) Password: SOLUTION
  - i) I used Excel Spreadsheet to illustrate the solution and mark specific pixels.
  - ii) I would go line by line and simply mark an X in the boxes needed.
  - iii) I used the Fill tool to highlight the X boxes and make the image more apparent.
  - iv) This seemed to be the fastest solution to me, as the grid is easy to understand and would be able to manipulate as needed.
  - v) I choose this solution over writing code to illustrate the pixels due to the speed of using Excel. Below is an image of my solution.



- 3) Monoalphabetic solution Caesar (15 pts)
  - a) Solution: ujqcsddessxsffes
  - b) un deux trois

- c) j'irai dans les bois
- d) Quatre cinq six
- e) Cueillir des cerises
- f) Sept huit neuf
- g) dans un panier neuf
- h) dix onze douze
- i) elles seront toutes rouges



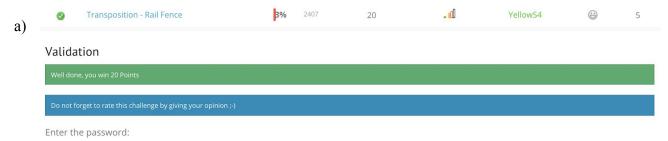
- My solution included writing java code that would shift the alphabet to decode the text. I soon realized that the shift did not equally apply to all of the words as one word would be french and the others still decoded and that each word was shifted by one more than the word to its right.
- m) I used a text document to record my attempts and then used my java code to ultimately find the solution.

```
public class Monoalphabeticsub {
public static void main(String[] args)
           String codedStr1 = "tm bcsv qolfp";
           String codedStr2 = "f'dmvd xuhm exl tgak";
           String codedStr3 = "hlrkiv sydg hxm";
           String codedStr4 = "qiswzzwf qrf oqdueqe";
           String codedStr5 = "dpae resd wndo";
           String codedStr6 = "liva bu vgtokx sjzk";
           String codedStr7 = "hmb rqch fqwbg";
           String codedStr8 = "fmmft seront sntsdr pmsecq";
           char[] alphabetArray = {'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'};
           char[] caesarArray = {'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'};
           for(int i = 0; i < codedStr1.length(); i++) {
                      if (codedStr1.charAt(i) != ' ') {
                                 for(int j = 0; j < caesarArray.length; j++) {
                                            if(codedStr1.charAt(i)==caesarArray[j]) {
                                                       System.out.print(alphabetArray[j]);
                      else {
                                 System.out.print(" ");
```

```
System.out.println();
for(int i = 0; i < codedStr2.length(); i++) {
          if (codedStr2.charAt(i) != ' ') {
                    for(int j =0; j < caesarArray.length; j++) {
                               if(codedStr2.charAt(i)==caesarArray[j]) {
                                         System.out.print(alphabetArray[j]);
          else {
                     System.out.print(" ");
System.out.println();
for(int i = 0; i < codedStr3.length(); i++) {
          if (codedStr3.charAt(i) != ' ') {
                    for(int j = 0; j < caesarArray.length; <math>j++) {
                               if(codedStr3.charAt(i)==caesarArray[j]) {
                                          System.out.print(alphabetArray[j]);
          else {
                     System.out.print(" ");
System.out.println();
for(int i = 0; i < codedStr4.length(); i++) {
          if (codedStr4.charAt(i) != ' ') {
                    for(int j = 0; j < caesarArray.length; <math>j++) {
                               if(codedStr4.charAt(i)==caesarArray[j]) {
                                          System.out.print(alphabetArray[j]);
          else {
                     System.out.print(" ");
System.out.println();
for(int i = 0; i < codedStr5.length(); i++) {
          if (codedStr5.charAt(i) != ' ') {
                    for(int j = 0; j < caesarArray.length; j++) {
                               if(codedStr5.charAt(i)==caesarArray[j]) {
                                          System.out.print(alphabetArray[j]);
          }
          else {
                     System.out.print(" ");
System.out.println();
for(int i = 0; i < codedStr6.length(); i++) {
```

```
if (codedStr6.charAt(i) != ' ') {
                     for(int j = 0; j < caesarArray.length; j++) {
                               if(codedStr6.charAt(i)==caesarArray[j]) {
                                         System.out.print(alphabetArray[j]);
          else {
                    System.out.print(" ");
System.out.println();
for(int i = 0; i < codedStr7.length(); i++) {
          if (codedStr7.charAt(i) != ' ') {
                    for(int j =0; j < caesarArray.length; j++) {
                               if(codedStr7.charAt(i)==caesarArray[j]) {
                                         System.out.print(alphabetArray[j]);
          else {
                     System.out.print(" ");
System.out.println();
for(int i = 0; i < codedStr8.length(); i++) {
          if (codedStr8.charAt(i) != ' ') {
                    for(int j = 0; j < caesarArray.length; <math>j++) {
                               if(codedStr8.charAt(i)==caesarArray[j]) {
                                         System.out.print(alphabetArray[j]);
          else {
                     System.out.print(" ");
System.out.println();
```

4) Transposition - Rail Fence



b) Password: Frozen Chicken

c) To solve this, I first researched rail fence in relation to cipher and realized that it was a cipher that involved writing the message down and up depending on the variation amount. I decided to use Excel to draw out the variations as I figured that this would be the quickest. I tried to layout some key words like invade and then match the pattern to that.

	A	В		С	D	E	F	G	1	Н	1	J	K	L	M	N	0	P	Q	R	S	Т	U	٧	W	X	Υ	Z	AA	AB	AC	AD	AE	AF	AG	AH	Al	AJ	AK	AL	AM	A١
1	Wn	b.r.iet	oeh	Fo'lk	Kutrts	znl (	cc hi e	e ekC	Otggs	nkid	y hin	i cna	neea	civo	lh																											
2	W																Ν														В											
3		1														Е		Т												0		Е										
4			L												K				U										Т				R									
5					L															С								С														H
6													E								K						0								Т						G	
7							1					D										Υ														Н				I		
8								N		F	١.														N												Е		E			
9									٧															0																		
10																																										
11																																										
12																																										
13																																										