Question 1) You obtain a file containing leaked passwords from Sony. One of the usernames is root and the associated password is 19f96fe30289e1ecc18a7a8504f750ea (if you are using C or Java, the hashed password is b524a4fc63f21233549a83d59113c410). You know that Sony does not use the latest security practices and uses MD5 to hash their password. Moreover, Sony limits the password length to 8 characters (upper case, lower case, and digits). Can you guess what the password is? How long did your password cracking program take? How much memory? Include the configuration of your machine.

Hint: Writing a program to try all possible passwords is okay and expected. Provide a copy of your source code (if any).

**import** java.util.\*;

**import** java.math.BigInteger;

**import** java.security.\*;

**public** **class** BruteForce{

**static** String *answer*="";

**public** **static** **void** main(String[] args)

{

Scanner in=**new** Scanner(System.***in***);

**char** ar[]={ 'P','A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K','L', 'M', 'N', 'O', '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'};

/\* char ar[]={ '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'};\*/

String enc;

System.***out***.print("Enter MD5 encrypted text : ");

enc=in.nextLine();

//HERE, 20 denotes the maximum wordlength 20

**final** **int** MAX\_WORDLENGTH = 8;//YOU JUST NEED TO CHANGE THIS TO MODIFY THE MAXIMUM WORDLENGTH

**for**(**int** wordlength = 8; wordlength <= MAX\_WORDLENGTH; wordlength++)

{

**if**(*generate*(wordlength,ar,enc))

{

System.***out***.print("Match found!! The decrypted string is : "+ *answer*);

**break**;

}

**else**

{

System.***out***.println("Not a word of "+wordlength+" characters");

}

}

}

**private** **static** **boolean** generate(**int** wordlength, **char**[] alphabet,String enc)

{

**final** **long** MAX\_WORDS = (**long**) Math.*pow*(alphabet.length, wordlength);

//System.out.println("MAX\_WORDS "+MAX\_WORDS);

**final** **int** RADIX = alphabet.length;

//System.out.println("RADIX "+RADIX);

**for** (**long** i = 0; i < MAX\_WORDS; i++)

{

**int**[] indices = *convertToRadix*(RADIX, i, wordlength);

**char**[] word = **new** **char**[wordlength];

**for** (**int** k = 0; k < wordlength; k++)

{

word[k] = alphabet[indices[k]];

}

String ss=**new** String(word);

/\*System.out.println("The String ss "+ss);\*/

**if**(*compareit*(*encrypt*(ss),enc))

{

*answer*=ss;

**return** **true**;

}

}

**return** **false**;

}

**private** **static** **int**[] convertToRadix(**int** radix, **long** number, **int** wordlength)

{

**int**[] indices = **new** **int**[wordlength];

**for** (**int** i = wordlength - 1; i >= 0; i--)

{

**if** (number > 0)

{

**int** rest = (**int**) (number % radix);

number /= radix;

indices[i] = rest;

}

**else**

{

indices[i] = 0;

}

}

**return** indices;

}

**public** **static** String encrypt(String str)

{

//String tryWord= "user1234";

**byte**[] msgByte = str.getBytes();

**try**{

MessageDigest msgDigest = MessageDigest.*getInstance*("MD5");

**byte**[] msgByteDigest = msgDigest.digest(msgByte);

BigInteger bigIntDigest = **new** BigInteger(1,msgByteDigest);

String digestedMessage = bigIntDigest.toString(16);

/\*System.out.println(digestedMessage);\*/

**return** digestedMessage;

}**catch**(NoSuchAlgorithmException ex){

ex.printStackTrace();

**return** **null**;

}

}

**public** **static** **boolean** compareit(String s2, String s1)

{

String a=s1;

**if**(s1.contains(s2))

**return** **true**;

**else**

{

/\*Java often misses out some zeroes while encrypting text, so here

\* I'm removing zeroes one by one from the original string and then

\* performing the check again\*/

**while**(a.indexOf('0')!=-1)

{

a=a.substring(0,a.indexOf('0'))+a.substring(a.indexOf('0')+1,a.length());

**if**(a.contains(s2))

**return** **true**;

}

}

**return** **false**;

}

}

Answer took approximately 16 days to run and crack the result :

real 22487m22.646s

user 0m0.000s

sys 0m0.000s

The cracked password is : POGS7EKo