**Example Questions for Final Exam**

**1-** Write a **procedure** that takes a string from the user and prints the text step by step.

**Example:**

Input: computer

Output:

c  
co  
com  
comp  
compu  
comput  
compute  
computer

**2-** Write a **function** that determines a given matrix is an *anti-identity matrix* or not.

An ***anti-identity matrix*** is a [matrix](http://en.wikipedia.org/wiki/Matrix_(mathematics)) where all the entries are zero except those on the diagonal being 1 and going from the lower left corner to the upper right corner (↗).

**Example:**



{{0,0,0,0,1},{0,0,0,1,0},{0,0,1,0,0},{0,1,0,0,0},{1,0,0,0,0}}

Output: true

**3-** Write a **function** that determines whether a given number is ***abundant*** or not.

An *abundant number* is a number for which the sum of its proper divisors is greater than the number itself.

For example; 12 is the first abundant number. Because its proper divisors are 1, 2, 3, 4 and 6 for a total of 16.

The first few abundant numbers are: 12, 18, 20, 24, 30, 36, 40, 42, 48, 54, 56, 60, 66, 70, 72, 78, 80, 84, 88, 90, 96, 100, 102, 104, 108, 112, 114, 120, ...

**Examples:**

Input: 112 Input: 14

Output: abundant Output: not abundant

**4-** Write a **function** that finds the majority character in the string if it exists.

The **majority character** in the **string, S**, of **size *n*** is a character that appears **more** **than n/2 times**.

**Example:**

Input: annem de evde deeeeeeeeeeeermişim

Output: e

**5-** Write a **function** that determines a given matrix is a ***signature matrix*** or not.

A ***signature matrix*** is a diagonal matrix whose diagonal elements are plus or minus 1.

A=\begin{pmatrix}
\pm 1   & 0       & \cdots & 0       & 0      \\
0       & \pm 1   & \cdots & 0       & 0      \\
\vdots  & \vdots  & \ddots & \vdots  & \vdots \\
0       & 0       & \cdots & \pm 1   & 0      \\
0       & 0       & \cdots & 0       & \pm 1  
\end{pmatrix}

**Example:**

{{1,0,0,0,0},{0,1,0,0,0},{0,0,-1,0,0},{0,0,0,-1,0},{0,0,0,0,1}}

Output: true

**6-** Write a **function** that *merges* two strings and returns the result in a new string.

**Example:**

string str1 = "deryaa";

string str2 = "hoca";

string new\_str = "";

Output: new\_str = "dheorcyaaa";

**7-** Write a **procedure** that repeatedly generates two random numbers and lets the user enter the sum of them until the answer is wrong.

**Example:**

2 + 10

12

correct

3 + 17

20

correct

8 + 6

16

Wrong

**8-** Write a **procedure** that implements ***SPLIT*** command.

Note: Don’t use split command in C#.

**Example:**

string str = "He said that it is not a good idea"

char separator = ‘ ‘

Output: He

said

that

it

is

not

a

good

idea

**9-** Write a **procedure** that generates a random number from 1 to 100. It asks the user to guess the number. The program outputs the message

“Your number is greater than my number”,

“Your number is less than my number” or

“You guessed the number”.

The program stops when the user finally guesses the number

**10-** Write a C# **procedure** that compares the student answers and correct answers of a ***multiple choice exam*** and then finds:

* The number of true answers
* The number of wrong answers
* The number of empty answers

**Example:**

string student\_answers = "AA BDBCACC";

string correct\_answers = "ADDCDBCACB";

Output: 6 true, 3 wrong, 1 empty answer

**11-** Write a C# **function** that determines whether a matrix is a ***tri-diagonal matrix*** or not.

A *tri-diagonal matrix* is a square matrix with nonzero elements only on the diagonal and slots horizontally or vertically adjacent the diagonal.

**Examples:**

1 3 0 0 0 1 8 0 0 0 0

1 4 1 0 0 2 1 3 0 0 0

0 2 2 1 0 0 2 4 1 0 0

0 0 5 3 1 0 0 4 6 5 0

0 0 0 2 4 0 0 0 6 1 3

0 0 0 0 2 7

Output: True

Output: True

int[,] a = new int[,] {{1, 3, 0, 0, 0},

{1, 4, 1, 0, 0},

{0, 2, 2, 1, 0},

{0, 0, 5, 3, 1},

{0, 0, 0, 2, 4}};

**12-** Write a program to automate ***Pig Latin*** translation.



A word game often played by elementary age children is to speak in “*Pig Latin*”. *Pig Latin* is abusive of the English language. In *Pig Latin*, every word that begins with a constant is transformed into a hyphenated word with the first portion coming from the original word less its first character and the portion after the hyphen is the first letter followed by “ay”. For Example, “pig” is transformed into “ig-pay” and “latin” into “atin-lay”.

The program should output the Pig Latin equivalent after each word. Note that word beginning with the vowels “a”, “e”, “i”, “o”, or “u” remain unchanged.

**Example:**

string str = "This program is supposed to automate the translation Input to the program will be several English sentences"

Output:

"his-Tay rogram-pay is upposed-say o-tay automate he-tay ranslation-tay Input o-tay he-tay rogram-pay ill-way e-bay everal-say English entences-say"

**13-** Write a **function** that determines a given matrix is a ***skew-symmetric matrix*** or not.

**Note**: Assign the array in the code. Don’t take from the user or don’t generate randomly.

A *skew-symmetric matrix* is a square matrix A whose transpose is also its negative; that is, it satisfies the equation A = −AT.  If the entry in the *i* th row and *j* th column is *aij*, i.e. *A* = (*aij*) then the symmetric condition becomes *aij* = −*aji*.

\begin{bmatrix}
0 & 2 & -1 \\
-2 & 0 & -4 \\
1 & 4 & 0\end{bmatrix}.

int[,] a = new int[,] {{0, 2, -1},

{-2, 0, -4},

{1, 4, 0}};

**14-** Write a program that *sums* two given ***clocks*** in *minute* and *second* unites, and prints the result. The variables should be **structured**.

**Examples:**

29' 13″ + 8' 16″ = 37' 29″

14' 40″ + 24' 25″ = 39' 5″