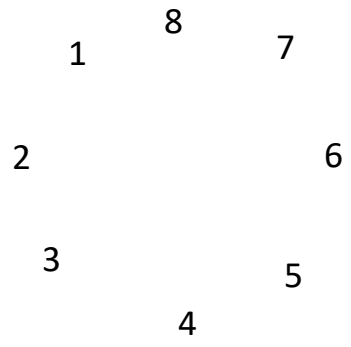
COSC 1437 (DL) - Fall 2016
Program Set #6

See 1437 Grading /Program Guide Sheet for directions and grading/submission information.

1. Groups of children often play games where one kid is designated as "it." Choosing this kid is often done by saying a phrase while pointing to the next kid for each word. The person managing the process, kid #1, starts with her/himself and starts the pointing in a counterclockwise fashion. When the phrase ends, the kid pointed at leaves the ring. The process is repeated until only one kid is left in the ring, who is declared it. For example, consider the phrase:

```
Eeny, meeny, miny, moe,
Catch a tiger by the toe.
If he hollers, let him go,
Eeny, meeny, miny, moe.
```



and 8 kids standing in the ring as shown in the figure to the right. After the first round, kid 4 is allowed to leave. After the second round, kid 2 is allowed to leave. After several more rounds, kid 8 is declared as it.

Write a C++ program to speed up the task of selecting who's "it." Input the number of children in the game, which will be between 1 and 500. The output should show the order of removal and should contain the number of the kid designated as "it." Check for invalid values. A user-defined function must be used. Output should be user friendly.

Name the program: `EenyMeenyXX.cpp`, where XX are your initials.

2. Write a C++ program that generates and traverses mazes. Two functions must be used:
   - One named `mazeGenerate` takes as an argument a two-dimensional array representing a randomly produced maze up to size 25. The function should also provide the starting and ending locations of the maze. The + symbol represents the walls of the maze, and periods(.) represent the squares in the paths through the maze. For each maze, the entrance and exit are along the left and right sides, excluding the corners (the top and bottom rows are solid walls).

```
+ + + + + + + + + +
+ + . + . . . . + +
+ + + . . . + + + +
+ . + . . . . + . .
. . . + . . . . . +
+ . . . . + . . + +
+ . . . + . + . . +
+ + . + + . . . + +
+ . + . . . . . . +
+ + + + + + + + + +
```

- The other function `mazeTraverse` must be recursive in nature. The function will receive a two-dimensional array representing the maze and the starting location of the maze. As `mazeTraverse` attempts to locate the exit from the maze, it should place the character `@` in the path. The function should display the maze after each move as the user can watch as the maze is solved.

```
  + + + + + + + + + + + +
  + . . + . + . . + + . +
  + . + + . . . . . . + +
  . . . + . + . . + . . +
  + + + . . . . . + . . @
  + + . . . . . . + . . +
  + . . . . . + . . . . +
  + . . + . + + . + . + +
  + . + . . . . + + + . +
  + . + . . . . + + + . +
  + + . + . . + . . . . +
  + + + + + + + + + + + +

Hit return to see next move

  + + + + + + + + + + + +
  + . . + . + . . + + . +
  + . + + . . . . . . + +
  . . . + . + . . + . . +
  + + + . . . . . + . @ @
  + + . . . . . . + . . +
  + . . . . . + . . . . +
  + . . + . + + . + . + +
  + . + . . . . + + + . +
  + . + . . . . + + + . +
  + + . + . . + . . . . +
  + + + + + + + + + + + +

Hit return to see next move
```

Output should be user friendly.

Name the program: `MazeProgramXX.cpp`, where XX are your initials.

3. Write a C++ program that prompts the user to input a string (a word) and then outputs the string in Pig Latin form. The program must store the characters of a string into a linked list. The rules for converting a string into Pig Latin form are as follows:

- If the string begins with a vowel, add the string `"-way"` at the end of the string. For example, the Pig Latin form of the `"eye"` is `"eye-way"`.
- If the string does not begin with a vowel, first add `"-"` at the end of the string. Then rotate the string one character at a time; that is, move the first character to the string to the end of the string until the first character of the string becomes a vowel. Then add the string `"ay"` at the end. For example, the Pig Latin form of the string `"There"` is `"ere-Thay"`.
- Strings such as `"by"` contain no vowels. In cases like this, the letter `y` can be considered a vowel. So, for this program the vowels are `a, e, i, o, u,` and `y` and their capital letter equivalents. Therefore, the Pig Latin form of `"by"` is `"y-bay"`.

- String such as "1234" contain no vowels. The Pig Latin form of the string "1234" is "1234-way". That is, the Pig Latin form of a string that has no vowels in it is the string followed by the string "-way".

Must use linked lists and C++ strings for full credit. Output should look similar to below.

Sample Run:

```
Enter a string:  by

The string you entered is: by
The Pig Latin form of the string: y-bay
```

Name the program: `LinkedPigLatinXX.cpp`, where XX are your initials.

4 (**). Write a C++ program to count pixels on a grid. The data are in a two-dimensional grid of cells, each of which may be empty (0) or filled (1). The filled cells are connected to form a blob (an object). A blob can consist of one or more ones connected horizontally, vertically, or diagonally. Thus a file consisting of all ones represents a single (very large) blob. The file begins with a line containing a single non-negative integer N that indicates the (square) dimensions of the grid (N x N). The value of N will never exceed 50. The next N lines will contain N characters. Output will consist of the file read in, the size of each blob in ascending order and the total number of blobs in the picture. If a grid contains no blobs, output the phrase: No blobs exist. Input the file name from the keyboard and check for valid files. Output should look similar to below.

Sample Input File:

```
15
000001111010000
000000100010000
100000010010001
000000011100000
000000000000000
000011000000000
000000000000000
001111000001000
000110000001000
000000100001000
000001100001100
000000100001111
000001111111111
000010000001111
000100000001111
```

Sample Run:

```
Enter the file name: blob.txt

00000111101 0000
0000001000 10000
1000000100 10001
0000000111 00000
0000000000 00000
0000110000 00000
0000000000 00000
0011110000 01000
0001100000 01000
0000001000 01000
0000011000 01100
0000001000 01111
0000011111 11111
0000100000 01111
0001000000 01111
```

```
There are 6 blobs on the grid.
Sizes: 1, 1, 2, 6, 12, 33
```

The sample above with the six areas highlighted. The input file will have spaces between the numbers. This program must be solved without using recursion, otherwise full credit will not be given.

Name the program: `BlobCounterXX.cpp`, where XX are your initials.