Sample Problems from Op 2003

By Eric Durant, PhD <durant@msoe.edu>
Monday 8 December 2003

General Information

More difficult problems are worth more points. The easiest problems usually consist of basic calculations and if/else logic. The medium level problems usually make use of functions and iteration (simple and nested). Array (or matrix, etc.) usage is sometimes necessary for the medium and high point value problems. The highest point value problems are usually only solved by a few teams, and they usually focus on basic data structures and text file processing. In 2004, each team may choose to use either Java or C++.

Easter (10 Points)

Calculate and display the date and month of Easter (*e.g.*, "April 20") based on user input of the 4+-digit year (*e.g.*, "2003") using the following formula.

$$
\begin{aligned}
G &= y \bmod 19 \\
C &= \frac{y}{100} \\
H &= \left(C - \frac{C}{4} - \frac{8C+13}{25} + 19G + 15\right) \bmod 30 \\
I &= H - \frac{H}{28}\left(1 - \frac{29}{H+1} \times \frac{21-G}{11}\right) \\
J &= \left(y + \frac{y}{4} + I + 2 - C + \frac{C}{4}\right) \bmod 7 \\
L &= I - J \\
\mathrm{m} &= 3 + \frac{L+40}{44} \\
\mathrm{d} &= L + 28 - 31 \times \frac{m}{4}
\end{aligned}
$$

…where "mod" is the modulo or integer remainder operator. Also, all divisions are ***integer divisions***.

y (year) is entered by the user. *m* and *d* represent the month and day found by the algorithm. All other (capitalized) variables represent intermediate values calculated by the algorithm. Dates and months are counted from 1. Easter is always on a Sunday in March (3) or April (4). This algorithm was published by Claus Tøndering in "Frequently Asked Questions about Calendars"

Most of North America switched to the modern (Gregorian) calendar, for which the given formula for Easter is valid, in September 1752. Therefore, your program must print an error message and perform no further calculations if the user requests that Easter be calculated prior to 1753.

Tic-tac-toe Calculator (20 Points)

Write a function whose argument is a 2-dimensional **`char`** array representing the current board state of a game of tic-tac-toe. The return value of the function will be the **`char`** 'x' if the x player won, 'o' if the o player won, and ' ' (space) if neither player won. You may assume that the input represents a reachable (valid) board state.

You may use a 2-D **`vector`** or an **`apmatrix`** instead of a 2-D array if you prefer.

Guess the Animal (40 Points)

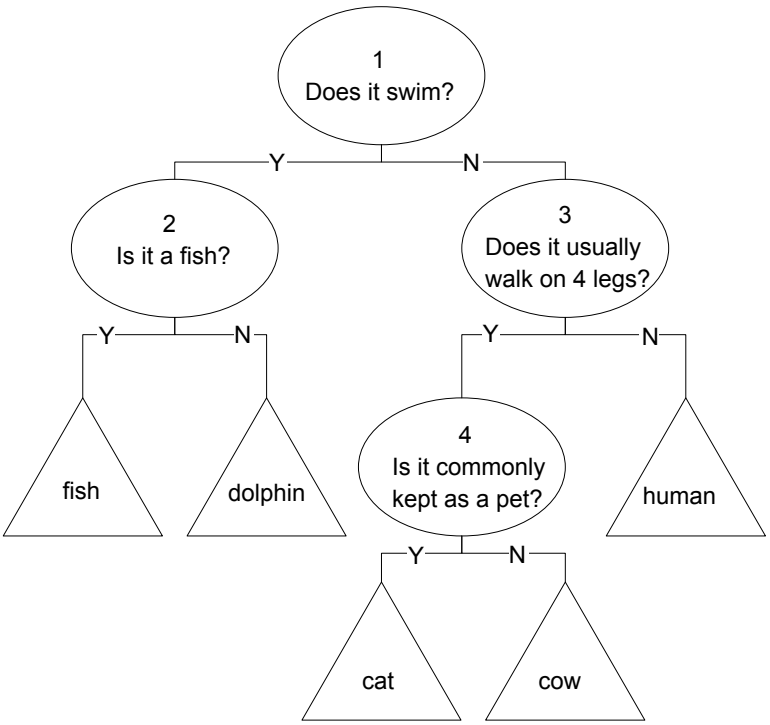
Write a program that plays "Guess the animal" with the user. The user will think of an animal and the computer will ask a number of yes-no questions of the user. The answers either lead the computer to ask another question or to give its best guess at the user's animal. To allow the computer's knowledge to be easily expanded, the information about the questions and answers must be stored in a file like the following…

```
1
Does it swim?
2
3

2
Is it a fish?
fish
dolphin

3
Does it usually walk on 4 legs?
4
human

4
Is it a commonly kept as a pet?
cat
cow
```



The file is divided into groups of 5 lines…

- 1st line: question number
- 2nd line: question
- 3rd line: what to do if the answer is "yes" – A number indicates that that question should be asked. Anything else is treated as the computer's best guess; a non-number means that the computer has run out of questions.
- 4th line: what to do if the answer is "no" (or "n", or "N" – be sure to tell the user if you require answers in a particular format)
- 5th line: blank

It might help to think of the information as forming a tree (see above).