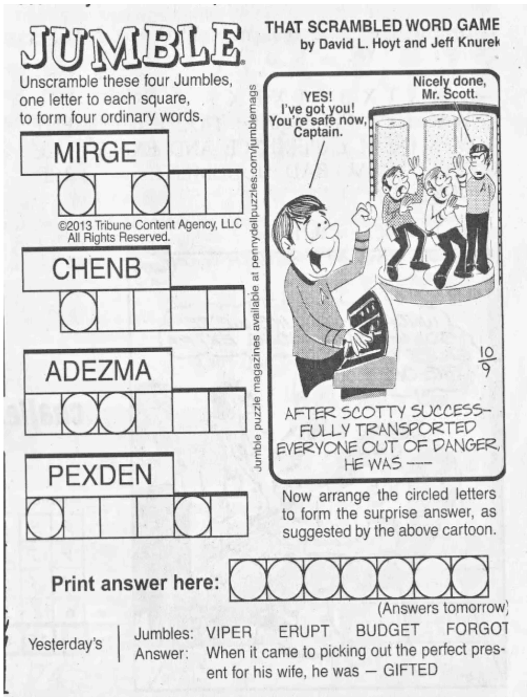
Assignment 1: Fortran Programming (25%)

WORD JUMBLE

An anagram is a type of word play, the result of rearranging the letters of a word or phrase to produce a new word or phrase. For example the anagram of **tea** is tea, tae, eat, eta, aet, ate. Anagrams can be used to solve word jumble problems. If a user provides a series of jumbled letters, all anagrams of the letters can be generated and then checked to see if they exist in a dictionary. The anagrams appearing in the dictionary are printed as potential solutions to the puzzle.



JUMBLE

THAT SCRAMBLED WORD GAME
by David L. Hoyt and Jeff Knurek

Unscramble these four Jumbles, one letter to each square, to form four ordinary words.

MIRGE

©2013 Tribune Content Agency, LLC
All Rights Reserved.

CHENB

ADEZMA

PEXDEN

Jumble puzzle magazines available at pennydellpuzzles.com/jumblemags

AFTER SCOTTY SUCCESSFULLY TRANSPORTED EVERYONE OUT OF DANGER, HE WAS ---

Now arrange the circled letters to form the surprise answer, as suggested by the above cartoon.

Print answer here:

(Answers tomorrow)

Yesterday's | Jumbles: VIPER ERUPT BUDGET FORGOT
Answer: When it came to picking out the perfect present for his wife, he was — GIFTED

TASK

Design and implement a Fortran program (`solvejumble`) which solves the individual word jumbles, producing a dictionary verified word. A data structure of some form will be needed to store the jumbled words and their solutions.

This program should contain, at minimum, the following series of subroutines:

1. A subroutine, `inputJumble()`, to obtain user input for the set of jumbles, i.e. jumbled letters `nelir`, `gurpe`.

2. A subroutine, `generateAnagram()`, to generate the anagrams from the letters for each jumbled word. The subroutine can use a recursive (or non-recursive if you like) function to generate anagrams from the jumbled string.

3. A subroutine, `findAnagram()`, to search for each of the anagrams in a dictionary, printing all result found.

The program `solvejumble` will use a **module** `lexicon`, that deals with the dictionary used by `findAnagram()`. The module will contain, at minimum, the following subroutines:

1. A subroutine, `buildlexicon()`, which builds a dictionary to search for the anagram in, from the system dictionary (this should be done once for each session).
    - On OSX the dictionary file on OS/X is located in `/usr/share/dict/words`.
    - Use a data structure of some form to store the words.
2. A subroutine `findlex()`, which search the lexicon for a particular word (the generated anagram). It will return true if the anagram is actually a word (as opposed to another jumble of letters).

NOTES:

- You may include any other subroutines you deem appropriate beyond those cited above.
- The program should be written in Fortran 95 or 2003.

Hint: Search for information on permutations.

COMPILING

Please do not include a Makefile, and make sure your program compiles in the following manner:

```
> gfortran -Wall solvejumble.f95 lexicon.f95
```

REFS

- Knight, D.G., "Anagrams and recursion", *Teaching Mathematics and its Applications*, Vol.5(3), pp.138-140 (1986).
- Morton, M., "Recursion + data structures = Anagrams", *BYTE*, Vol.12(13), pp.325-334 (1987).

USER INTERFACE

The program can run in one of two ways:

- Enter all the jumbled words as a group, and then process them as a group, or
- Enter each jumbled word one at a time, and process them individually.

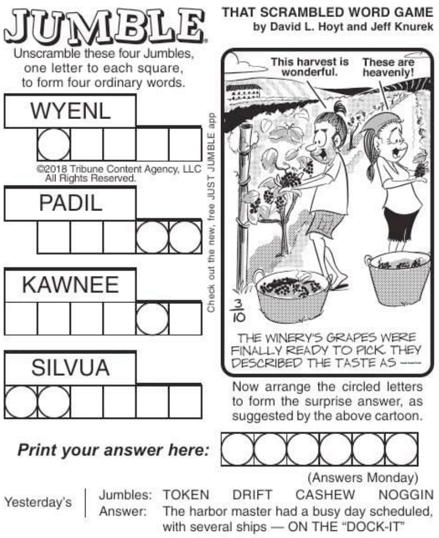
Here is a **sample** of what the program **might** look like for the puzzle above, using a block input. It requires manual input of the circled letters of the solved words to generate the final anagram.

```
How many jumbled words? 4

Enter the 4 jumbled words:
>MIRGE
>CHENB
>ADEZMA
>PEXDEN

The following jumbles have been solved:
MIRGE     grime
CHENB     bench
ADEZMA    amazed
PEXDEN    expend

Solve word jumble puzzle?
Y

Select circled letters from word puzzle:
grime: g i
bench: b
amazed: m a
expend: e n
Jumbled word: gibmaen

Solved jumble: beaming
```

Note, you may also decide to enter the circled positions in the puzzle as you enter each word. This means they could be automatically selected from the solved jumbles. For example:

```
How many jumbled words? 4
Enter the jumbled word:
>MIRGE
Enter the circled positions: 1 3
>CHENB
Enter the circled positions: 1
```

TESTING



JUMBLE

THAT SCRAMBLED WORD GAME
by David L. Hoyt and Jeff Knurek

Unscramble these four Jumbles,
one letter to each square,
to form four ordinary words.

WYENL
○□□□□

PADIL
□□□○○

KAWNEE
□□□□○□

SILVUA
○○□□□□

Check out the new, free JUST JUMBLE app

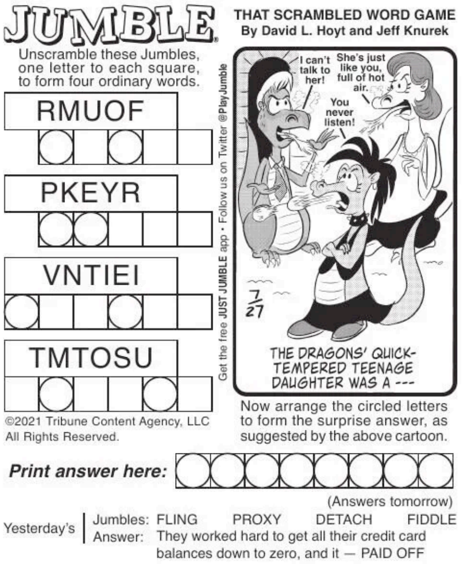
THE WINERY'S GRAPES WERE FINALLY READY TO PICK. THEY DESCRIBED THE TASTE AS ---

Now arrange the circled letters to form the surprise answer, as suggested by the above cartoon.

Print your answer here: ○○○○○○

(Answers Monday)

Yesterday's | Jumbles: TOKEN DRIFT CASHEW NOGGIN
Answer: The harbor master had a busy day scheduled, with several ships — ON THE "DOCK-IT"



JUMBLE

THAT SCRAMBLED WORD GAME
By David L. Hoyt and Jeff Knurek

Unscramble these Jumbles,
one letter to each square,
to form four ordinary words.

RMUOF
○□○□□

PKEYR
○○□□□

VNTIEI
○□□○□□

TMTOSU
□○□□○□

Get the free JUST JUMBLE app • Follow us on Twitter @PlayJumble

THE DRAGONS' QUICK-TEMPERED TEENAGE DAUGHTER WAS A ---

Now arrange the circled letters to form the surprise answer, as suggested by the above cartoon.

Print answer here: ○○○○○○○○

(Answers tomorrow)

Yesterday's | Jumbles: FLING PROXY DETACH FIDDLE
Answer: They worked hard to get all their credit card balances down to zero, and it — PAID OFF

CASE 1:

WYENL ⟶ newly
PADIL ⟶ plaid
KAWNEE ⟶ weaken
SILVUA ⟶ visual

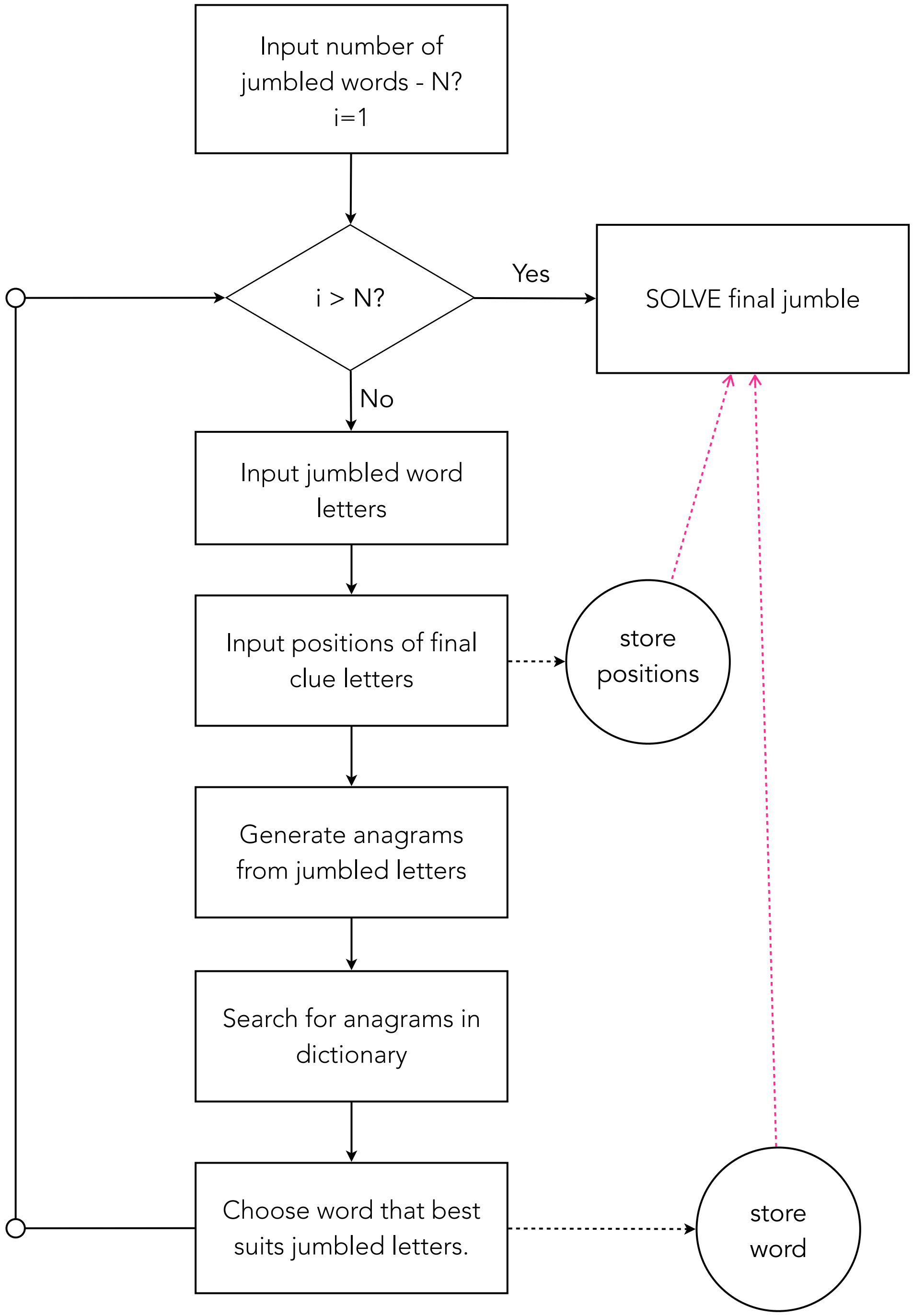
Letters for final jumble: **n**ewly, pla**id**, weak**e**n, **vi**sual ⟶ n, i, d, e, v, i
Jumble solved = **divine**

CASE 2:

RMUOF ⟶ forum
PKEYR ⟶ perky
VNTIEI ⟶ invite
TMTOSU ⟶ utmost

Letters for final jumble: **f**o**r**um, **pe**rky, **i**nv**i**te, u**t**mo**s**t ⟶ f, r, p, e, i, i, t, s
Jumble solved = **spitfire**

APPENDIX: Flow chart for complete "Jumble" solver


Input number of jumbled words - N?
i=1
i > N?
Yes
SOLVE final jumble
No
Input jumbled word letters
Input positions of final clue letters
store positions
Generate anagrams from jumbled letters
Search for anagrams in dictionary
Choose word that best suits jumbled letters.
store word

ASSIGNMENT INFORMATION

REFLECTION REPORT

Describe your Fortran program in one (1) page (single spaced) **reflection report**, explaining decisions you made in the process of designing the program. Consider the design document a synopsis of your programming process. One page should include a synopsis of the approach you took to design the program (e.g. it could be a numbered list showing each step in the process).

Some of the questions that should be answered include:

- Was Fortran well suited to solving the problem?
- What particular structures made Fortran a good choice?
- Given your knowledge of programming, was Fortran easy to learn?

DELIVERABLES

The submission should consist of the following items:

- The reflection report (PDF).
- The code (well documented and styled appropriately of course).
  `solvejumble.f95 lexicon.f95`

- Both the code and the reflection report can be submitted as a ZIP, TAR, or GZIP file.

STYLING & COMMENTS

Style consists of mnemonic variable names, indentation, and the use of whitespaces and paragraphing. The purpose of good style is to make the meaning of your program clear to someone who has never seen it before, cannot run it, and cannot talk with you. Documentation consists of in-code documentation. Examples of qualities to look for include:

- Are variable names well chosen?
- Are comments relevant rather than simple repetitions of the code?
- Do comments point out key sections of code, indicate special cases, or make assertions?
- Are the indents 3 or 4 spaces? Do not use tabs or 2 space indenting (please check "convert tabs to spaces" in your editor)
- Is whitespacing used to separate parts of the program to provide clarity?

SKILLS

- Fortran programming, ability to write a Fortran program, ability to work with Fortran modules, ability to work with appropriate data structures.