Computer Programming

Bachelor in Biomedical Engineering
Bachelor in Applied Mathematics and Computing
Course 2020 / 2021

Exercise Sheet 12

Recursion, Input/Output, …

- SOLUTIONS -

Content Table

Exercise 1

Ulam stated that starting from any integer number, if you follow these steps you'll reach number 1:

- If the number is even divide it by 2
- If the number is odd multiply it by 3 and add 1

Write a recursive function to prove Ulam's conjecture. The program asks the user to introduce a number and prints on screen all the numbers obtained until it reaches 1.

SOLUTION

```
function recUlam(num)
if num == 1
    fprintf('%d\n', num);
else
    if rem(num,2) == 0
        newnum = num/2;
    else
        newnum = num*3+1;
    end
    if newnum ~= 1
        fprintf('%d\n', newnum);
    end
    recUlam(newnum);
end
end
```

Exercise 2

Normally, numbers are represented using the decimal system (base 10). It is said that the decimal system is a positional system, since the value of each digit depends on its relative position. For example:

$792 = 7 * 10^2 + 9 * 10^1 + 2 * 10^0$

Write a recursive function that transforms a number from the decimal system base representation to another base. Additionally, write a main program that asks the user to introduce a number and its new base, and that prints the correspondent value in that new base on screen.

To transform the number follow the following steps:

- Successively divide the number by its new base
- The remainders obtained will be the digits of the new number
- The process ends when we cannot divide the number anymore as its current value is 0

Example:

Introduce the number to transform: 8
Introduce the new base: 2

The new value is 1000

Note: $1000 = 1*10^3+0*10^2+0*10^1+0*10^0$

Example:

Introduce the number to transform: 6
Introduce the new base: 2

The new value is 110

Note: $110 = 1*10^2+1*10^1+0*10^0$

MAIN PROGRAM

```
value = input('Introduce the number to transform: ');
base = input('Introduce the new base: ');
rdo = changebase(value,base);
fprintf('The new value is %d\n', rdo);
```

FUNCTION

```
function [value] = changebase(num,base)
if num ~= 0
    vcoef = floor(num/base);
    vrem = rem(num,base);
    value = vrem+ 10*changebase(vcoef,base);
else
    value = 0;
end
end
```

Exercise 3

Write a function 'reverseVector' that receives a vector and, using recursion, returns a vector containing the elements in reverse order.

SOLUTION

```
function [revVect] = reverseVector(vect)
% Function reverseVector receives a vector and returns a vector
% containing the same elements in reverse order
maxLength = length(vect);
```

```
if maxLength == 1
    revVect = vect;
else
    revVect = [vect(maxLength) reverseVector(vect(1:maxLength-1))];
end
end
```

Exercise 4

Write a program for maintaining a phone list. The program uses a text file named 'phones.txt' to store the information about the phone numbers and the person they belong to. The format of the file is:

name phonenumber

where both *name* and *phonenumber* are strings (let's consider you might need to store international numbers including the character '+')

The program should allow the user to perform the following operations:

- introduce a new phone number: this operation requires the user to introduce the name of the person and his/her phone number
- modify the number of a person: this operation requires the user to introduce the name of the person which exists in the file and his/her new phone number
- list all the phone numbers
- display the phone number of a given person
- exit the program

Note: the program should create the file if it doesn't exist already. When the user finishes the program (exit) it should store all the phone numbers in the file, so the next time he/she runs the program this information can be retrieved.

Example:

```
1. Insert a new phone number
2. Modify existing person's phone number
3. List all
4. List a person's phone number
5. Exit (save to file)

Enter option: 1

Enter the name of the person: Baggins
Enter the new phone number: +34647564234
Press any key to show menu

1. Insert a new phone number
2. Modify existing person's phone number
3. List all
4. List a person's phone number
```

```
5. Exit (save to file)

Enter option: 1

Enter the name of the person: Bolson
Enter the new phone number: +34565673422
Press any key to show menu

1. Insert a new phone number
2. Modify existing person's phone number
3. List all
4. List a person's phone number
5. Exit (save to file)

Enter option: 3

Name    Phone
Baggins         +34647564234
Bolson          +34565673422

Press any key to show menu

1. Insert a new phone number
2. Modify existing person's phone number
3. List all
4. List a person's phone number
5. Exit (save to file)

Enter option: 2

Enter the name of the person whose number you want to change: Baggins
Baggins's number is +34647564234
What should the new number be? (empty to erase) +34647564666
Baggins' number has been overwritten with value +34647564666
Press enter to show menu

1. Insert a new phone number
2. Modify existing person's phone number
3. List all
4. List a person's phone number
5. Exit (save to file)

Enter option: 4

Whose phones do you want to know? Bolson
Bolson's number is: +34565673422

Press enter to show menu

1. Insert a new phone number
```

```
2. Modify existing person's phone number
3. List all
4. List a person's phone number
5. Exit (save to file)

Enter option: 5

File saved. Bye!
```

Content of the file *phones.txt* after execution:

```
Baggins +34647564666
Bolson  +34565673422
```

SOLUTION

See exercise 5.

Exercise 5

Modify the previous program to allow to store up to three phone numbers per person. The format of the file will now be:

name phonenumber1-phonenumber2-phoneumber3

Note that some of the contacts in your phone list might have only one number, while other might have two or three.

Example content of the file:

```
Baggins 647564234-456876780-345634567
Bolson 345656734 --
```

FUNCTION

```
function [index] = indexByName(phoneList,nameToLookFor)
% looks for nameToLookFor in phoneList struct array and returns the index
% of the first occurrence or -1 if not found
% phoneList struct has field .name
bFound = 0;
i = 1;
while (bFound == 0) && i <= length(phoneList)
    if strcmpi(phoneList(i).name, nameToLookFor) ==1
        bFound = 1;
    else
        i = i+1;
    end
end
% if found, i now holds the index of the found occurrence since the while
loop
% stopped as soon as we found the name, and i did not increment after
that.
if bFound
    index = i;
```

```
else
    index = -1;
end
end
```

MAIN PROGRAM

```
clear;
clc;
filename = 'phones.txt';
% First, open the file to be able to read the list from the file into a
struct array
fid = fopen(filename, 'rt');
if (fid ~= -1) % No error opening file (file already exists)
    % Store all file content into a struct
    % Then, modify this struct
    % When finished, overwrite the file with the new struct
    %
    % The struct has two fields:
    %   phoneBook(i).name   stores the name of the person
    %   phoneBook(i).numbersCA  stores a cellArray of phonenumbers
    %   (length can vary from 1 to 3 phonenumbers)
    % e.g.
    %   phoneBook(1).name   Baggins'
    %   phoneBook(1).numbersCA  {{'666110000'},{'666001122'}}

    i = 0;
    while(feof(fid) == 0)
        i = i + 1;
        clear phoneCA;
        phoneCA = textscan(fid,'%s %s', 1);
        phoneBook(i).name = phoneCA{1}{1};
        numbersToSplit = phoneCA{2}{1};

        clear auxCA
        auxCA = {};
        [auxCA{1}, rest] = strtok(numbersToSplit,'-');
        [secondNumber, rest] = strtok(rest,'-');
        if isempty(secondNumber) == 0
            auxCA{2} = secondNumber;
            [thirdNumber, rest] = strtok(rest,'-');
            if isempty(thirdNumber) == 0
                auxCA{3} = secondNumber;
            end
        end
        phoneBook(i).numbersCA = auxCA;
    end
    fclose(fid);
else % If file did not exist
    phoneBook = [];
end

disp('1. Insert a new phonenumber');
disp('2. Modify existing person''s phonenumbers');
disp('3. List all');
disp('4. List a person''s phonenumbers');
disp('5. Exit (save to file)');
fprintf('\n');
option = input('Enter option: ');
```

```
fprintf('\n');
while (isempty(option) == 1) || option < 1 || option > 5
    option = input('Enter a valid option: ');
end
while (option ~= 5)
    switch(option)
        case 1 % Insert a new phonenumber (and possibly a person)
            name = input('Enter the name of the person: ', 's');
            % Check if the person is currently in the list or not
            i = indexByName(phoneBook,name);
            if i ~= -1
                alreadyPresent = 1;
            else
                alreadyPresent = 0;
            end
            if alreadyPresent == 1
                % Check if we can add a new phone number to the existing
                %person
                clear auxCA;
                auxCA = phoneBook(i).numbersCA;
                if length(auxCA) < 3 % At least one space to add a phone
number
                    newPhonenumber = input('Enter the new phonenumber:
','s');
                    while (isempty(newPhonenumber) == 1)
                        newPhonenumber = input('Enter a valid new
phonenumber: ','s');
                    end
                    % Add number at the end of this person's phone number
list
                    auxCA{length(auxCA)+1} = newPhonenumber;
                    phoneBook(i).numbersCA = auxCA;
                    fprintf('Number %s added to %s''s phone number
list\n', newPhonenumber, name);
                else
                    fprintf('Sorry, %s''s phone list is full\n', name);
                end
            else % Not already present (new name)
                newPhonenumber = input('Enter the new phonenumber:
','s');
                while (isempty(newPhonenumber) == 1)
                    newPhonenumber = input('Enter a valid new
phonenumber: ','s');
                end
                clear auxCA;
                auxCA{1} = newPhonenumber;
                newIndex = length(phoneBook) + 1;
                phoneBook(newIndex).name = name;
                phoneBook(newIndex).numbersCA = auxCA;
            end
        case 2 % Modify an existing person's phone number list
            nameToLookFor = input('Enter the name of the person whose
number you want to change: ', 's');
            i = indexByName(phoneBook,nameToLookFor);
            % Index returns either the index of the first occurrence or -
1
            % if not found
            if (i ~= -1) % If found
                auxCA = phoneBook(i).numbersCA;
                if (length(auxCA) < 1)
                    fprintf('Sorry, %s does not have any number to
modify\n', nameToLookFor);
```

```
                else % At least one number to modify available
                    % Print available numbers
                    for j=1:length(auxCA)
                        fprintf('%s''s number %d is %s\n' ,
nameToLookFor, j, auxCA{j} );
                    end
                    numToChange = input('Which number do you want to
replace? ');
                    while (numToChange < 1) || (numToChange >
length(auxCA)) % Invalid number
                        numToChange = input('Invalid number. Which number
do you want to replace? ');
                    end
                    newNum = input('What should the new number be? (empty
to erase) ','s');
                    if (isempty(newNum) == 1) % Erase number
                        auxCA(numToChange) = []; % Note () for accessing
cell, not cell content
                        phoneBook(i).numbersCA = auxCA;
                        fprintf('%s''s number %d has been deleted\n',
nameToLookFor, numToChange);
                    else % Overwrite number
                        auxCA{numToChange}=newNum;
                        phoneBook(i).numbersCA = auxCA;
                        fprintf('%s''s number %d has been overwritten
with value %s\n', nameToLookFor, numToChange, newNum);
                    end
                end
            else % If name not found
                fprintf('Sorry, %s was not found in this list\n',
nameToLookFor);
            end
        case 3 % List all
            fprintf('Name\tPhone1\tPhone2\tPhone3\n');
            for i=1:length(phoneBook)
                fprintf('%s', phoneBook(i).name);
                auxNumbersCA = phoneBook(i).numbersCA;
                for j=1:length(auxNumbersCA)
                    fprintf('\t%s',auxNumbersCA{j});
                end
                fprintf('\n');
            end
        case 4 % List phone nubmers of a person
            % Ask for person's name and look for it
            % (list considered not ordered)
            nameToLookFor = input('Whose phones do you want to know?
','s');
            while(isempty(nameToLookFor) == 1)
                nameToLookFor = input('Invalid name. Whose phones do you
want to know? ','s');
            end
            i = indexByName(phoneBook,nameToLookFor);
            % index returns either the index of the first occurrence or
            % -1 if not found
            if (i ~= -1) % If found
                fprintf('%s''s numbers are:\n', nameToLookFor);
                auxNumbersCA = phoneBook(i).numbersCA;
                for j=1:length(auxNumbersCA)
                    fprintf('Number %d is %s\n', j, auxNumbersCA{j});
                end
                fprintf('\n');
            else % If not found
```

```
                fprintf('Sorry, %s was not found in this list\n',
nameToLookFor);
            end
            % Case 5 (exit) implemented after while loop
    end

    fprintf('\n');
    input('Press any key to show menu ');
    fprintf('\n');
    disp('1. Insert a new phonenumber');
    disp('2. Modify existing person''s phonenumbers');
    disp('3. List all');
    disp('4. List a person''s phonenumbers');
    disp('5. Exit (save to file)');
    fprintf('\n');
    option = input('Enter option: ');
    fprintf('\n');
    while (isempty(option) == 1) || option < 1 || option > 5
    option = input('Enter a valid option: ');
    end
end
% Case exit: dump list to file, overwriting it
outputfid = fopen(filename, 'wt'); % Empties file if it exists
if outputfid == -1
    disp('Error opening file for dumping info');
else
    for i=1:length(phoneBook)
        auxNumbersCA = phoneBook(i).numbersCA;
        fprintf(outputfid, '%s %s',phoneBook(i).name,auxNumbersCA{1});
        j = 2;
        while j <= length(auxNumbersCA)
            fprintf(outputfid, '-%s',auxNumbersCA{j});
            j = j + 1;
        end
        while j <= 3
            fprintf(outputfid, '-');
            j = j + 1;
        end
        fprintf(outputfid, '\n');
    end
    fclose(outputfid);
    disp('File saved. Bye!');
end
```

Exercise 6

Write a version of the game “Simon Says” for MATLAB. The program presents a sequence of numbers between 0 and 9, one by one. More specifically, MATLAB shows a number, pauses for 1 second, cleans the screen, shows another number and repeats the process. Then, the user is required to introduce the sequence of numbers. If the sequence of numbers introduced is correct, the next sequence of numbers of the game will be one element longer (start with a sequence that is 4 numbers long). If the user introduces an incorrect sequences three times the program finishes and displays the total number of sequences that the player has been able to reproduce.

Note:

- to clear the window you can use the command: `clc`
- to pause the execution one second use: `pause(1)`
- to obtain a random number between 0 and 9: `floor(rand()*10)`

Example:

```
**** New sequence ****
Press s when you are ready to start: s
Here, MALTAB will show the sequence of numbers 7 7 1 4 4 one by one, pausing one
second between numbers and cleaning the screen each time
Now introduce the values in the same order
Introduce a value: 7
Introduce a value: 7
Introduce a value: 1
Introduce a value: 4
Introduce a value: 4
Well done!

**** New sequence ****
Press s when you are ready to start: s
Here, MALTAB will show the sequence of numbers 6 7 7 2 6 6 one by one, pausing one second
between numbers and cleaning the screen each time
Now introduce the values in the same order
Introduce a value: 6
Introduce a value: 7
Introduce a value: 7
Introduce a value: 2
Introduce a value: 6
Introduce a value: 6
Well done!

**** New sequence ****
Press s when you are ready to start: s
Here, MALTAB will show the sequence of numbers 1 1 4 9 2 6 6 one by one, pausing one second
between numbers and cleaning the screen each time
Now introduce the values in the same order
Introduce a value: 1
Introduce a value: 1
Introduce a value: 4
Introduce a value: 9
Introduce a value: 2
Introduce a value: 6
Introduce a value: 6
You failed!
The right sequence was: 1 1 4 9 2 6 6
```

SOLUTION

```
clear;
clc;
clear numbers;
clear usnumbers;
vnumbs = 4;
misses = 0;
while (misses <3)
    disp('**** New sequence ****');
    sready = input('Press s when you are ready to start ', 's');
    while (sready ~= 's')
        sready = input('Press s when you are ready to start ', s');
    end
    for i=1:vnumbs
        clc;
        value = floor(rand()*10);
        fprintf('%d\n', value);
        numbers(i) = value;
        pause(1);
    end
    clc;
    disp('Now introduce the values in the same order');
    for j=1:vnumbs
        auxval = input('Introduce a value: ');
        while (isempty(auxval) == 1)
            auxval = input('Introduce the last value again: ');
        end
        usnumbers(j) = auxval;
    end
    if usnumbers == numbers
        disp('Well done!');
        vnumbs = vnumbs + 1;
    else
        disp('You failed!');
        disp('The right sequence was: ');
        for i=1:vnumbs
            fprintf('%d ',numbers(i));
        end
        misses = misses + 1;
        fprintf('You have %d more attempts\n', 3-misses);
    end
end
fprintf('End of the game. You were able to reproduce %d sequences\n',
vnumbs-4);
```