CS2002   Tutor:Marwan Fayed   170025298

W01 – Practical Report

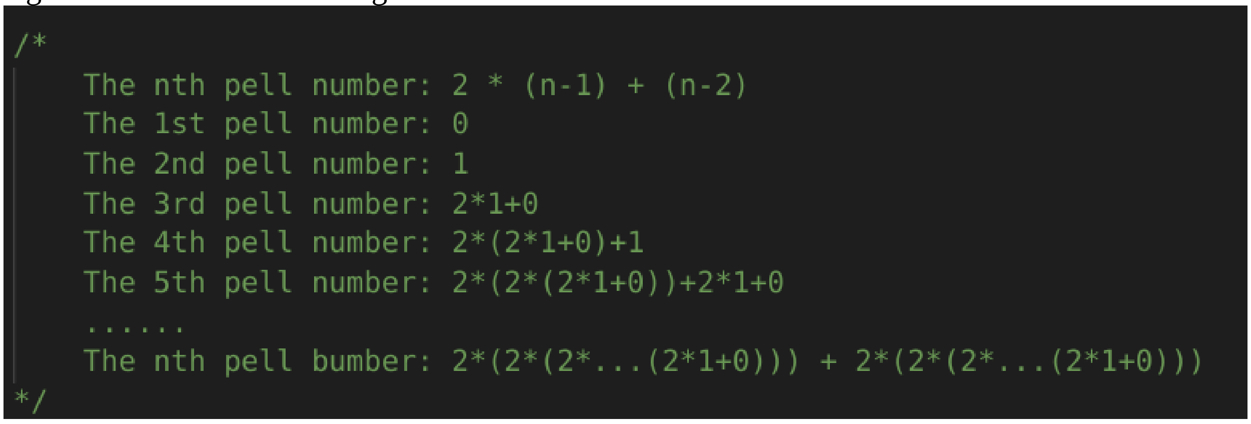
Content:

Design:

All programs are designed in order to fit the requirement of the practical and five files and one makefile available in this folder.

Pell (Main function): Since C is a function oriented programming language, it is hard to turn a value from one file to another file. So the nth pell number will be calculated in this method instead of being finished in Recursive function or iterative function.

Stage 1: Recursive function: Since logic has been noted in the recursive file, and the design of this program also based on such logic



```
/*
    The nth pell number: 2 * (n-1) + (n-2)
    The 1st pell number: 0
    The 2nd pell number: 1
    The 3rd pell number: 2*1+0
    The 4th pell number: 2*(2*1+0)+1
    The 5th pell number: 2*(2*(2*1+0))+2*1+0
    ......
    The nth pell bumber: 2*(2*(2*...(2*1+0))) + 2*(2*(2*...(2*1+0)))
*/
```

So the (n-1) and (n-2) number are being recursively calculated until n = 1 and n = 0 and the pell number returns 1 and 0.

Stage 2: Iterative function: Iterative function will be calculated using the same logic as recursive function. The difference is that three variables are defined in this method to save the result, before ~~→ (n-1)~~ and beforeplus→(n-2).

Stage 3: Personalised function: This function requires user input the initial first and second value and length of the result (show result from 1st to nth value). This function needs a new main method for user input and check the such input. The logic of the personalised function is as same as iterative method.

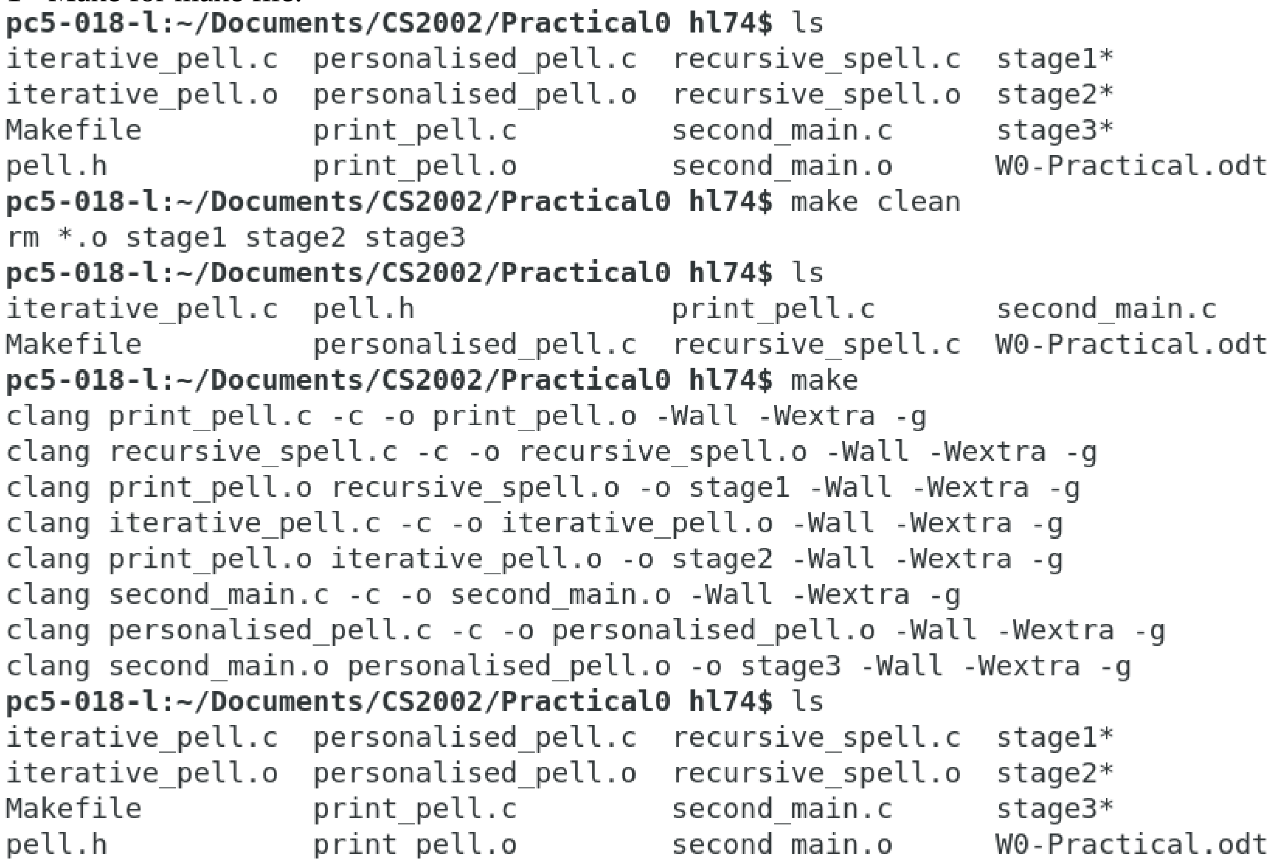
Implementation:

Stage 1: In this stage only ensuring when n is 0 then the program will return 0 and when n is 1 the program will return 1 and then recursively get the result.

Stage 2: In this stage, n-1 and n-2 and result are being defined in function as integers in order to temporary save the values and calculate from 1st pell number to nth pell number.

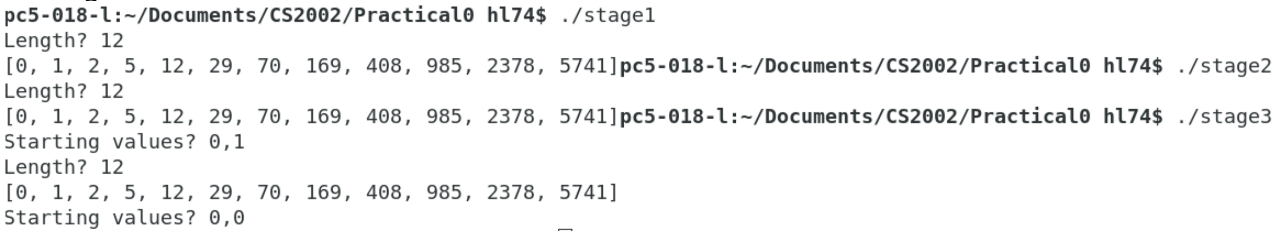
Stage 3: In this stage, two initial values of pell numbers when n = 0 and n = 1 are defined as two extern values in order to read inputs from main method and pass them to personalised.c file. Since the program should ensure that two initial values will not being changed, so I set two integers and make them equal to initial values such that initial values will not be changed.

Testing:
Several tests are being done for this file
1> Make for make file:



```
pc5-018-l:~/Documents/CS2002/Practical0 hl74$ ls
iterative_pell.c  personalised_pell.c  recursive_spell.c  stage1*
iterative_pell.o  personalised_pell.o  recursive_spell.o  stage2*
Makefile          print_pell.c         second_main.c      stage3*
pell.h            print_pell.o         second_main.o      W0-Practical.odt
pc5-018-l:~/Documents/CS2002/Practical0 hl74$ make clean
rm *.o stage1 stage2 stage3
pc5-018-l:~/Documents/CS2002/Practical0 hl74$ ls
iterative_pell.c  pell.h               print_pell.c       second_main.c
Makefile          personalised_pell.c  recursive_spell.c  W0-Practical.odt
pc5-018-l:~/Documents/CS2002/Practical0 hl74$ make
clang print_pell.c -c -o print_pell.o -Wall -Wextra -g
clang recursive_spell.c -c -o recursive_spell.o -Wall -Wextra -g
clang print_pell.o recursive_spell.o -o stage1 -Wall -Wextra -g
clang iterative_pell.c -c -o iterative_pell.o -Wall -Wextra -g
clang print_pell.o iterative_pell.o -o stage2 -Wall -Wextra -g
clang second_main.c -c -o second_main.o -Wall -Wextra -g
clang personalised_pell.c -c -o personalised_pell.o -Wall -Wextra -g
clang second_main.o personalised_pell.o -o stage3 -Wall -Wextra -g
pc5-018-l:~/Documents/CS2002/Practical0 hl74$ ls
iterative_pell.c  personalised_pell.c  recursive_spell.c  stage1*
iterative_pell.o  personalised_pell.o  recursive_spell.o  stage2*
Makefile          print_pell.c         second_main.c      stage3*
pell.h            print_pell.o         second_main.o      W0-Practical.odt
```

Since all stages are being made using make and make clean will clean all objects.
2> stage test:

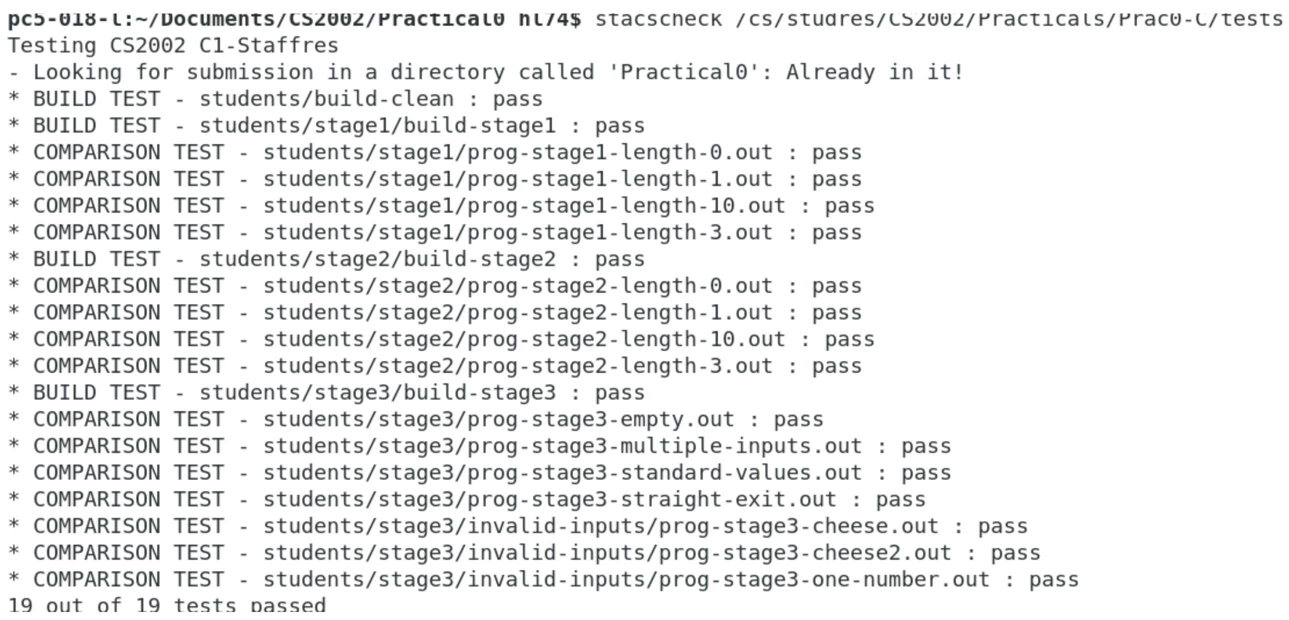


```
pc5-018-l:~/Documents/CS2002/Practical0 hl74$ ./stage1
Length? 12
[0, 1, 2, 5, 12, 29, 70, 169, 408, 985, 2378, 5741]pc5-018-l:~/Documents/CS2002/Practical0 hl74$ ./stage2
Length? 12
[0, 1, 2, 5, 12, 29, 70, 169, 408, 985, 2378, 5741]pc5-018-l:~/Documents/CS2002/Practical0 hl74$ ./stage3
Starting values? 0,1
Length? 12
[0, 1, 2, 5, 12, 29, 70, 169, 408, 985, 2378, 5741]
Starting values? 0,0
```

I controlled the length of pell numbers and test whether they will get the same results, if results are same then the test passed, or the result will failed.

4> Problem facing:
Several problems I have faced in this practical:
1. Argument control: Since when users input values that are not acceptable numbers, the program will return Invalid Inputs by checking the values are 0.
2. Amount of arguments control: Since inputs may not being fully input and the program will return invalid input by checking the scanf() is not equal to the amount of variables that being inputted.

5> Stacscheck:



```
pc5-018-l:~/Documents/CS2002/Practical0 hl74$ stacscheck /cs/studres/CS2002/Practicals/Prac0-C/tests
Testing CS2002 C1-Staffres
- Looking for submission in a directory called 'Practical0': Already in it!
* BUILD TEST - students/build-clean : pass
* BUILD TEST - students/stage1/build-stage1 : pass
* COMPARISON TEST - students/stage1/prog-stage1-length-0.out : pass
* COMPARISON TEST - students/stage1/prog-stage1-length-1.out : pass
* COMPARISON TEST - students/stage1/prog-stage1-length-10.out : pass
* COMPARISON TEST - students/stage1/prog-stage1-length-3.out : pass
* BUILD TEST - students/stage2/build-stage2 : pass
* COMPARISON TEST - students/stage2/prog-stage2-length-0.out : pass
* COMPARISON TEST - students/stage2/prog-stage2-length-1.out : pass
* COMPARISON TEST - students/stage2/prog-stage2-length-10.out : pass
* COMPARISON TEST - students/stage2/prog-stage2-length-3.out : pass
* BUILD TEST - students/stage3/build-stage3 : pass
* COMPARISON TEST - students/stage3/prog-stage3-empty.out : pass
* COMPARISON TEST - students/stage3/prog-stage3-multiple-inputs.out : pass
* COMPARISON TEST - students/stage3/prog-stage3-standard-values.out : pass
* COMPARISON TEST - students/stage3/prog-stage3-straight-exit.out : pass
* COMPARISON TEST - students/stage3/invalid-inputs/prog-stage3-cheese.out : pass
* COMPARISON TEST - students/stage3/invalid-inputs/prog-stage3-cheese2.out : pass
* COMPARISON TEST - students/stage3/invalid-inputs/prog-stage3-one-number.out : pass
19 out of 19 tests passed
```

6> Options: The program achieved reject invalid inputs and negative inputs.