Scientific Computing

Lab Assignment 2

Due: Friday, Sept. 21, 2018 at 17:00

Refer to week 1 lecture notes for instructions for submitting your assignment using the handin webpage. In your submission include

- all required source code and
- any other files asked for in the question.

Put the files in a directory called `lab-2`. Do not include object or executable files.
For full marks document your work using *meaningful* comments.
Make sure to include

- your name,
- your student number, and
- the assignment number.

in each file you submit.
Also add comments where necessary to clearly label each solution.

---

1. Type in the Name and age example from section 3.3 (or the course slides from week 2) into a file you name `name_and_age.cpp`:

```
int main(){
  cout << "Please enter your first name and age" <<endl;
  string first_name;
  int age;
  cin >> first_name;
  cin >> age;
  cout << "Hello "<< first_name<<" (age "<<age<<")"<<endl;
  return 0;
}
```

   Get it to compile. Add a second string `last_name` and another >> operator to read it in. Modify the code to ask for an age in years as a `double` and print it out in months by multiplying by 12.

2. Work through the "repeated words" example from Stroustrup (pp.71-72):

```
int main(){
  string previous = " ";
  string current;
  while (cin >> current){
    if (previous == current) {
      cout << "repeated word :"
        << current << endl;
              }
      previous = current;
  }
  return 0;
}
```

   (a) Type the code into `RepeatedWords.cpp`, and add line-by-line comments to describe what each line is doing. Get it to compile and run.

   (b) In a text-file `RepeatedWords.txt`, walk through the code line-by-line describing what is in memory if the user of the program inputs "The cat cat jumped"

3. Implement the `square(x)` function without using the * operator – instead add x to the return value x times. Test your function in a simple main function that prints a table of squares from 0 to 100, in a file `MultTable.cpp`.

4. (Exercise 4 from Ch. 4) Write a program to play a numbers guessing game in a file `GuessingGame.cpp`. The user thinks of a number between 1 and 100 and your program asks questions to figure out what the number is (eg. "Is the number you are thinking of less than 50?"). Your program should be able to identify the number after asking no more than seven questions. Hint: Use the < and <= operators and the `if-else` construct.

5. (Exercise 5 from Ch. 4) Write a program that performs as a very simple calculator in a file `SimpleCalculator.cpp`. Your calculator should be able to handle the four basic math operations – add, subtract, multiply, and divide – on two input values. Your program should prompt the user to enter three arguments: two `double` values and a character to represent an operation.

6. Include a makefile that builds all of the programs above.