Prof. Dr. Karsten Urban
Mazen Ali
16.04.2018
Summer 2018
Sheet 0

Numerical Finance - C++ warmup

You can compile your program example.cpp as follows:

```
g++ example.cpp -o example -Wall
```

Afterwards, you can run it calling

```
./example
```

1. **Prerequisites:** Basic program structure, constant variables, char pointer, strings, output.

   - Write a program that outputs "Hello, World!" by printing a **const char *** with value "Hello, World!".
   - Write a program that outputs "Hello, World!" by printing a **string** with value "Hello, World!".

2. **Prerequisites:** Basic program structure, for loops, while loops, output.
   Write a program that outputs "Hello, World" n times (where n is a nonnegative integer that the user will input) with:

   - a **for** loop
   - a **while** loop
   - a **do ... while** loop

3. **Prerequisites:** input/output, modulus operator, loops.
   Write a program to read a number N from the user and then find the first N primes. A prime number is a number that only has two divisors, one and itself.

4. **Prerequisites:** input/output, modulus operator, loops, aborting loops, ternary operator.
   Write a program that loops indefinitely. In each iteration of the loop, read in an integer N (declared as an int) that is inputted by a user, output N5 if N is nonnegative and divisible by 5, and -1 otherwise. Use the **ternary operator (?:)** to accomplish this. (Hint: the modulus operator may be useful.)

   a) Modify the code from so that if the condition fails, nothing is printed. Use an **if** and a **continue** command (instead of the ternary operator) to accomplish this.

   b) Modify the code to let the user break out of the loop by entering -10 or any negative number. Before the program exits, output the string "Goodbye!".

5. Do this problem without the use of a computer! What does this snippet do? Try doing out a few examples with small numbers on paper if you're stuck.

```
// bob and dole are integers
int accumulator = 0;
while (true){
    if(dole == 0) break ;
    accumulator += ((dole % 2 == 1) ? bob : 0);
    dole /= 2;
    bob *= 2;
}
cout << accumulator << endl;
```

6. **Prerequisites:** Variable types and range.
   Here is the code for a factorial program. Copy it and verify that it compiles.

```
#include <iostream>

using namespace std;

int main(){
    short number;
    cout << "Enter a number: ";
    cin >> number;

    cout << "The factorial of " << number << " is ";
    int accumulator = 1;
    for (; number > 0; accumulator *= number--);
    cout << accumulator << endl;

    return 0;
}
```

   a) What do you get when you enter the following values: 0, 1, 2, 9, 10?

   b) Run the program and enter -1. What happens? How can you change the code such that it won't exhibit this behavior?

   c) Try entering some larger numbers. What is the minimum number for which your modified program stops working properly? (You shouldn't have to go past about 25). Can you explain what has happened?

   d) Modify the given code such that negative inputs do not break the program and the smallest number that broke the program before now works. Do not hardcode answers.

7. **Prerequisites:** input/output, loops, if/else conditions, random numbers.
   Write a program that calculates a random number 1 through 100. The program then asks the user to guess the number. If the user guesses too high or too low then the program should output "too high" or "too low" accordingly. The program must let the user continue to guess until the user correctly guesses the number.

a) Modify the program to output how many guesses it took the user to correctly guess the right number.

b) Modify the program so that instead of the user guessing a number the computer came up with, the computer guesses the number that the user has secretely decided. The user must tell the computer whether it guesed too high or too low.

c) Modify the program so that no matter what number the user thinks of $(1 - 100)$ the computer can guess it in 7 or less guesses.

8. The following snippet has bugs. Identify them and indicate how to correct them. Do this without the use of a computer!

```
class Point{
 private :
   int x, y;

 public :
   Point (int u, int v) : x(u),y(v){}

   int getX () { return x; }
   int getY () { return y; }

void doubleVal (){
   x *= 2;
   y *= 2;
}

};

int main (){
   const Point myPoint(5, 3);
   myPoint.doubleVal();

   cout << myPoint.getX() << " " << myPoint.getY() << endl;
   return 0;
}
```

9. **Prerequisites:** input/output, loops, if/else conditions, random numbers, arrays.
Make a two player tic tac toe game.

a) Modify the program so that it will announce when a player has won the game (and which player won, x or o)

b) Modify the program so that it is a one player game against the computer (with the computer making its moves randomly)

c) Modify the program so that anytime the player is about to win (aka, they have 2 of 3 x's in a row, the computer will block w/ an o)

10. **Prerequisites:** input/output, loops, if/else conditions, random numbers, arrays, classes, constructors,...
Write a program that creates a list of bunny objects. Each bunny object must have

- Sex: Male, Female (random at creation 50/50)
- color: white, brown, black, spotted
- age : 0-10 (years old)
- Name : randomly chosen at creation from a list of bunny names.
- vampire bunny: true/false (decided at time of bunny creation 2% chance of true)

Rules:

- At program initialization 5 bunnies must be created and given random colors.
- Each turn afterwards the bunnies age 1 year.
- So long as there is at least one male age 2 or older, for each female bunny in the list age 2 or older; a new bunny is created each turn. (i.e. if there was 1 adult male and 3 adult female bunnies, three new bunnies would be born each turn)
- New bunnies born should be the same color as their mother.
- If a bunny becomes older than 10 years old, it dies.
- If a vampire bunny is born then each turn it will change exactly one non radioactive bunny into a vampire bunny. (if there are two vampire bunnies two bunnies will be changed each turn and so on...)
- Vampire bunnies are excluded from regular breeding and do not count as adult bunnies.
- Vampire bunnies do not die until they reach age 50.

The program should print a list of all the bunnies in the colony each turn along with all the bunnies details, sorted by age. The program should also output each turns events such as

```
Bunny Thumper was born!
Bunny Fufu was born!
Vampire Bunny Darth Maul was born!
Bunny Julius Caesar died!
```

The program should write all screen output to a file. When all the bunnies have died the program terminates. If the bunny population exceeds 1000 a food shortage must occur killing exactly half of the bunnies (randomly chosen).

- Modify the program to run in real time, with each turn lasting 2 seconds, and a one second pause between each announement.