CSCM12 – Software concepts and efficiency February 24th 2025
Manjiri Joshi, Cécilia Pradic & Deshan Sumanathilaka

Lab 4: more recursion!

- For sign-off, if the exercise is code, you are expected to show one working copy of the code. Otherwise, write down a worked solution (i.e., as you would in an exam) to show us.
- The questions marked as **Challenge** are going beyond what we expect from you. Kudos if you manage them, but they won't count towards extra marks on the coursework.

1. **Bubble sort** Let us try to introduce our first algorithm to sort arrays. This one is known as *bubble sort*. You can use either recursion or loops (or mix them) to solve this exercise.

   (a) Write a function

   ```
   static void bubbleDownStep(int[] arr, int i)
   ```

   that takes as input an array a and an index i of that array, and that swaps the elements of index i and $i+1$ if $a(i) > a(i+1)$. This function should run in constant time ($\mathcal{O}(1)$).

   (b) Deduce a function

   ```
   static void bubbleDown(int[] arr, int i)
   ```

   that takes as input an array a and an index i of that array, and assuming that `arr` is sorted up to index i, applies `bubbleDownStep` a number of times so that `arr` is sorted up to index $i+1$. This function should run in linear time ($\mathcal{O}(n)$).

   (c) Deduce a function

   ```
   static void bubbleSort(int[] arr)
   ```

   which sorts the array `arr`. What is its asymptotic complexity?

2. **Time complexity of recursive functions** For each of the following function, estimate its time complexity.

   (a)
   ```
   static int horner(ArrayList<Integer> p, int v)
   {
     if(p.size() <= 0)
       return 0;
     else
     {
       final int p0 = p.remove(p.size()-1);
       // you can assume that p.remove(p.size()) is O(1)
       return p0 + v * horner(p, v);
     }
   }
   ```

   (b)
   ```
   static int exp(double a, int n)
   {
     if(n == 0)
   ```

```
    return 1;
  int r = exp(n / 2);
  return (n % 2 == 0) ? r * r : r * r * a;
}
```

(c)
```
static double evalPoly(double[] p, double v)
{
  int n = p.length;
  double r = 0;
  for(int i = 0; i < n; ++i)
    r += p[i] * exp(v, i);
  return r;
}
```

3. **Tournaments** For this question, we will view instant elimination tournaments as a recursive algorithm to identify the best team.

   (a) Assume you have a class `Team` and a function `static boolean match(Team a, Team b)` which returns `true` if team a wins and `false` if team b wins (there are no draws). Write a recursive function of type

   `static Team tournamentWinner(Team[] arg)`

   in java that simulates a tournament and outputs the winner.

   (b) What is the time complexity of your solution? (use the master theorem to determine it)

   (c) Consider the following function.

```
static Team bestTeam(Team[] arg) {
   bestteam = arg[0];
   for (i=1,i++,i<arg.length) {
      if match(arg[i],bestteam) {
         bestteam = arg[i];
      }
   }
   return bestteam;
}
```

   Does it output the same thing as yours always? If not, provide a concrete type `Team`, a concrete function `match` and an array `arg` for which the two functions disagree.