Name: _____

IB Computer Science 2
CPU cache worksheet

Locality of reference

On modern computers, it takes orders of magnitude longer to access a single memory location than to execute a single instruction. Fortunately, programs tend to exhibit a property called **locality** that processors can exploit to hide this huge latency in the common case.

1. Consider the following code:
   ```
   int value = 0;
   ...
   ```

   Do you expect that the program is done accessing `value`? Why or why not?

2. Now consider this code:
   ```
   int value = 0;
   value = Integer.parseInt(console.readLine("Enter a value: "));
   ...
   ```

   Do you expect that the program is done accessing `value`? Why or why not?

The program property you just observed is known as **temporal locality**.

3. Consider the following code:
   ```
   int[] values = new int[8];
   values[0] = Integer.parseInt(console.readLine("1st value? "));
   ...
   ```

   Which index of `values` do you expect the program to access next? Why?

The program property you just observed is known as **spatial locality**.

4. In addition to data, program code is stored in memory. Which type of locality is more relevant to a program's instructions?

5. Given this code:

```
int sum = 0;
for(int i = 0; i != n; ++i) { sum += a[i]; }
```

Indicate the type(s) of locality exhibited by each of the program's variables.

Variable	Temporal locality?	Spatial locality?	No locality?
`i`	✔		
`a`			
`a[i]`			
`n`			
`sum`			

Cache

Modern processors contain a small on-chip memory called a **cache**. Because it is physically close to the data path, it is much faster to access.

6. Whenever the program accesses a memory location, the processor checks for a copy in cache. If there is one, it reads or writes only the cached copy. Otherwise, it copies the value from memory into cache. For which type of locality does this improve access time?

7. When the processor copies data from memory to cache, it does not retrieve only the few bytes that make up the requested value. Instead, it takes a large group of surrounding bytes called a **cache line**. For which type of locality does this improve access time?

8. Let's measure your CPU's cache lines, whose size will be a power of 2. Have one group member download `CacheEffects.java`. Simultaneously, someone else should create a spreadsheet and populate its first column with the numbers 1, 2, 4, 8, 16, and 32.

    a. Run the program several times, providing each of the above values. Record the resulting time measurements.
    b. Generate a scatterplot or line graph. How many `long`s fit in a cache line?
    c. Look up the size of a Java `long`. How many bytes is each cache line?
    d. Check your answer with the command: `sysctl -a | grep -F hw.cache`

If you have extra time…

Comment out the last array access in the measured loop. Start a second data series and collect new measurements. What do you notice? Look up **hardware prefetching** and decide which type of locality it relates to. Why is the line that you commented out key to our benchmark?

(Adapted from an activity co-authored for CMU 15-213 Introduction to Computer Systems)