Danh Nguyen
65872969 (danhn2)

Rodney McGrath
55352053 (rjmcgrat)

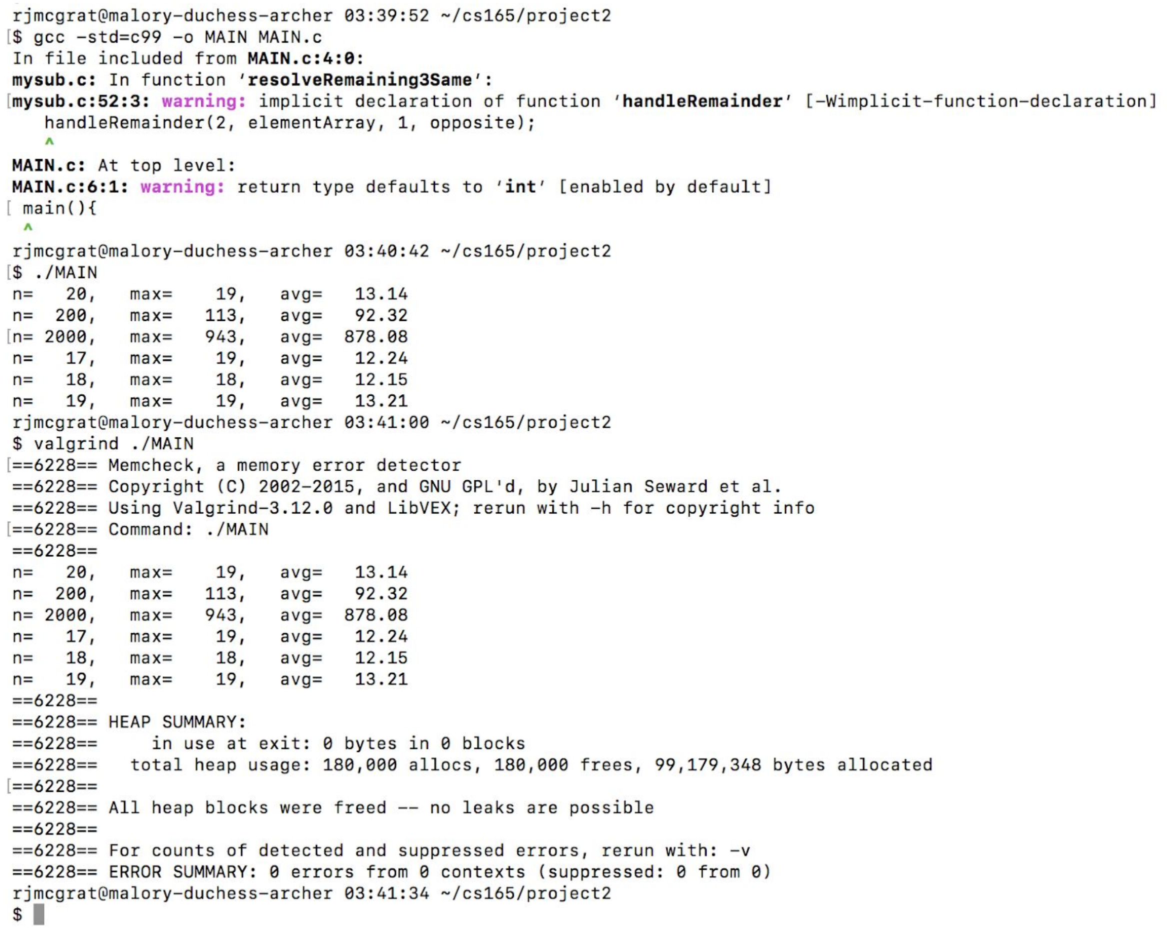
Assignment 2: Majority

Program is compiled by typing gcc -std=c99 -o MAIN MAIN.c and ran by typing ./MAIN

Data Structure and Implementation:

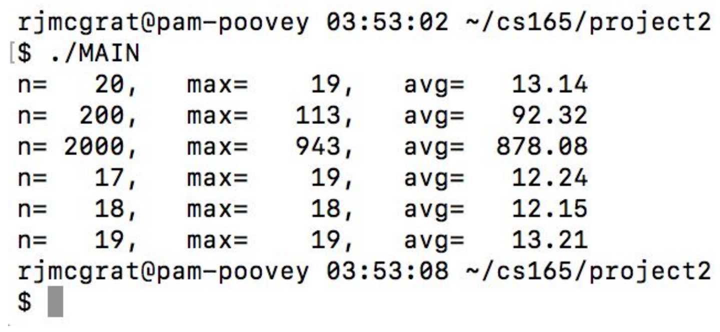
- There is no specific data structure for our algorithm. The main part of the algorithm is first figure out some reference indices and then use it to find the majority. The algorithm divides marbles into two sides. It is easy to find the majority marbles when the query result is 0 or 4. However, whenever, there is a query results which is 2, we store the current group of 4 marbles in an array in order to process it later. The later processing of the query result 2 groups will be done in pair. We essentially compare 2 different indices between 2 pairs of 3 same marble groups. This will then give us a query result of 0 in which we don't have to do anything and 4 in which it must be decided which side the marbles will go toward. Overall, it is when the query result returns a 2 which makes it troublesome.

Valgrind memory check:



```
 rjmcgrat@malory-duchess-archer 03:39:52 ~/cs165/project2
[$ gcc -std=c99 -o MAIN MAIN.c
 In file included from MAIN.c:4:0:
 mysub.c: In function 'resolveRemaining3Same':
[mysub.c:52:3: warning: implicit declaration of function 'handleRemainder' [-Wimplicit-function-declaration]
    handleRemainder(2, elementArray, 1, opposite);
    ^
 MAIN.c: At top level:
 MAIN.c:6:1: warning: return type defaults to 'int' [enabled by default]
[ main(){
  ^
 rjmcgrat@malory-duchess-archer 03:40:42 ~/cs165/project2
[$ ./MAIN
 n=   20,   max=    19,   avg=   13.14
 n=  200,   max=   113,   avg=   92.32
[n= 2000,   max=   943,   avg=  878.08
 n=   17,   max=    19,   avg=   12.24
 n=   18,   max=    18,   avg=   12.15
 n=   19,   max=    19,   avg=   13.21
 rjmcgrat@malory-duchess-archer 03:41:00 ~/cs165/project2
 $ valgrind ./MAIN
[==6228== Memcheck, a memory error detector
 ==6228== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
 ==6228== Using Valgrind-3.12.0 and LibVEX; rerun with -h for copyright info
[==6228== Command: ./MAIN
 ==6228==
 n=   20,   max=    19,   avg=   13.14
 n=  200,   max=   113,   avg=   92.32
 n= 2000,   max=   943,   avg=  878.08
 n=   17,   max=    19,   avg=   12.24
 n=   18,   max=    18,   avg=   12.15
 n=   19,   max=    19,   avg=   13.21
 ==6228==
 ==6228== HEAP SUMMARY:
 ==6228==     in use at exit: 0 bytes in 0 blocks
 ==6228==   total heap usage: 180,000 allocs, 180,000 frees, 99,179,348 bytes allocated
[==6228==
 ==6228== All heap blocks were freed -- no leaks are possible
 ==6228==
 ==6228== For counts of detected and suppressed errors, rerun with: -v
 ==6228== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
 rjmcgrat@malory-duchess-archer 03:41:34 ~/cs165/project2
 $ █
```

Result of functions:



```
rjmcgrat@pam-poovey 03:53:02 ~/cs165/project2
[$ ./MAIN
n=   20,   max=    19,   avg=   13.14
n=  200,   max=   113,   avg=   92.32
n= 2000,   max=   943,   avg=  878.08
n=   17,   max=    19,   avg=   12.24
n=   18,   max=    18,   avg=   12.15
n=   19,   max=    19,   avg=   13.21
rjmcgrat@pam-poovey 03:53:08 ~/cs165/project2
$ 
```

Empirical complexity analysis:

Observed worse case:

- 943 Queries for an array of 2000 marbles.

Average case:

- 878.08 Queries for an array of 2000 marbles.

Theoretical complexity analysis:

Theoretical Worst Case: O(n) or n/2.

At worse it takes n/4 queries initially. The for the absolute worse we have to assume that all groups of 4 marbles will yield a query results 2 and in order to further resolve such group of marbles will require at worse 3 additional comparisons. As a result, n/4 + 3 * n is O(n).

Theoretical Expected Worst Case: for an array of 2000 the expected worst case is 2.

Theoretical Average: O(m + n) or 1.25 * m + (n/4) + 8

m is the number of 2 query result groups.

Since we know that for a query result of 2 there is a .5 chances of only doing 1 queries. The case where 1 queries are used is when comparing some reference indices with indices of 11 or 00 are compared with the reference. The remaining .25 of the time will only requires for 1 additional queries to be used. Other .25 chance requires 2 additional queries. As a result, to make a decision/ determine the majority marbles in a group of 4 marbles (where the initial query result is 2) requires in total 1.25 multiply by the number of 2 query result groups. Also, at the beginning of the algorithm, we always requires n operations in order to categorize the different groups. So on average we requires 1.25 * m + n/4 queries, In addition, on average 5 queries are needed to find a reference array and 3 queries to resolve the six marbles within the first 10 marbles division.