```
/*
* @author Cole Van Verth
* @pengo cverth
* @email colevanverth@gmail.com
* @file main.cpp
* @assignment 4: Radix/Bucketsort Hybrid
*/

/**
* This program reads non-negative numbers from STDID, sorts them using a Radix
* Bucket sort hybrid algorithim, and then prints the number to STOUT in
* ascending order.
*
* The program compiles successfully with no warnings. Two major tests were
* performed on this program. First, the sort routine was tested to ensure that
* it had O(n) run time. The sort routine was timed with 10,000, 100,000 and
* 1,000,000 elements. The times varied by roughly 1000%, so it appeared that the
* routine was successful in achieving O(n) run time. Second, the sort routine
* was tested for correctness. 1,000,000 elements with numbers from 0 to
* 999,999,999 were tested and 1,000,000 elements with numbers from 0 to 9999
* were tested (high number of duplicates). The same data sets were sorted with
* the ‘sort’ command, then the outputs were compared with diff. Output was
* identical, so the test for correctness was successful. The edge case
* of zero elements was tested and did not crash the program. Lastly, the program
* was tested for memory leaks and none were found.
*/
```

?

Changes made since tests?

```
#include <iostream>
#include <iomanip>
#include <vector>

   /**
   * Sorts non-negative numbers up to 9 (base 10) digits long in acsending
   * order. Bucket are created for every hex digit (16). For the 8 hex digits
   * needed to represent the 9 (base 10) digits, the entries in ‘A’ are placed
   * into their corresponding hex digit bucket. The routine starts with the
   * least significant digit and iterates towards the most significant digit.
   * The routine then iterates over the buckets and places them back into ‘A’,
   * then repeats the process for the next digit.
   * @param ‘A‘ vector to sort containing non-negative integers up to 9 (base
   * 10) digits long
   */
void RadixBucketSort(std::vector<int> & A);

void RadixBucketSort(std::vector<int> & A) {
   // Sets up constants
   int digitRange = 16; // 16 digits (including zero) in hex
   int digitAmount = 8; // 8 hex digits needed to represent 9 base ten digits
   int numElements = A.size();

   // Creates a bucket for every (single) possible digit
   auto buckets = new std::vector<int>[digitRange];

   for (int i = 0; i < digitAmount; i++) {
      for (int j = 0; j < numElements; j++) {
         // Isolates the 4 bits representing the i’th digit of A[n] in hex
         unsigned int digit = A[j];
         digit = digit >> (i * 4); // Removes unneccesary bits to the right
         digit = digit << 24; // Removes unneccesary bits to the left (7 hexs)
         digit = digit >> 24; // Remaining 4 bits pos restored on right (7 hexs)
         buckets[digit].push_back(A[j]); // Places num onto corresponding bucket
```

crashing here — Check values for digit and j?

I suspect digit

```
      }
      int indexArray = 0;
      for (int j = 0; j < digitRange; j++) {
         for (auto number : buckets[j]) {
            // Data placed from buckets back into ‘A’
```

```
            A[indexArray] = number;
            indexArray++;
         }
         buckets[j].clear(); // Clears the buckets for next digit to be sorted
      }
   }
   delete[] buckets;
}

int main() {
   // Loads numbers from STDIN
   int numBuffer;
   std::vector<int> data;
   while (std::cin >> numBuffer) {
      data.push_back(numBuffer);
   }

   // Sorts the data
   RadixBucketSort(data);

   // Prints the sorted vector to STDOUT
   for (auto number : data) {
      std::cout << std::setfill('0') << std::setw(9) << number << std::endl;
   }
}
```

✓

int?

```
p4: main.o
        g++ -o p4 main.o

main.o: main.cpp
        g++ -c main.cpp

clean:
        rm -f p4 *.o *~
```

60/100

tests

Segfault (crash) on all test runs?

(see attached)

```
TEST RUNS
---------

steveh@pengo:~/cs21sp23-4/cverth$ ./p4 < 3M-r > temptemp
Segmentation fault (core dumped)
steveh@pengo:~/cs21sp23-4/cverth$ ./p4 < d100k
Segmentation fault (core dumped)
steveh@pengo:~/cs21sp23-4/cverth$ ./p4
1
2
3
4
5
15
25
35
45
115
125
215
314
313
132
terminate called after throwing an instance of 'std::bad_alloc'
  what():  std::bad_alloc
Aborted (core dumped)
steveh@pengo:~/cs21sp23-4/cverth$
```

Small manual test case