

**Arithmetic and Decision-making in C****Partitioning Digits of an Integer**

For this assignment, you will use very basic C techniques to implement a C function to remove from a given nonnegative integer all the even (or odd) digits, and return the resulting integer. For example:

```
{23410, EVENHIGH}  --> 24031
{23410, ODDHIGH}   --> 31240
{2640, ODDHIGH}    --> 2640
{1023712, EVENHIGH} --> 221371
```

You will provide an implementation for a C function that performs this calculation:

```
// Rather than use a nondescriptive label, or none at all, we will use
// an enumerated type to make the logic of of the code clearer:
enum Ordering {EVENHIGH, ODDHIGH};

/** Computes a new integer from N by separating all the even digits
 *  digits from N and all the odd digits from N.
 *
 *  For example:
 *  {23410, EVENFIRST}  --> 24031
 *  {23410, ODDFIRST}   --> 31240
 *  {2640, ODDFIRST}    --> 2640
 *  {1023712, EVENFIRST} --> 221371
 *
 *  Pre:  N is initialized
 *        Ordering is EVENHIGH or ODDHIGH
 *  Returns: integer obtained by reordering the digits of N as described
 *
 *  Restrictions:
 *  - uses only its parameters and local automatic variables
 *  (i.e., no global variables)
 *  - does not make any use of character variables or arrays
 *  - does not read input or write output
 */
uint32_t PartitionInteger(uint32_t N, enum Ordering order);
```

Make useful comments in your implementation of `PartitionInteger()`. The same general guidelines for commenting that you have been taught in your Java courses should provide sufficient guidance.

**Getting Started**

A tar file is available, containing the testing/grading code that will be used to grade your solution:

<code>driver.c</code>	test driver... read the comments!
<code>PartitionInteger.h</code>	header file declaring the specified function... do not modify!
<code>PartitionInteger.c</code>	C source file for implementing the specified function
<code>Generator.h</code>	header file declaring the test case generator... do not modify!
<code>Generator.o</code>	64-bit object file containing the test case generator
<code>checkAnswer.h</code>	header file declaring the result checking code... do not modify!
<code>checkAnswer.o</code>	64-bit object file containing the result checker

Download the tar file `C01Files.tar` from the course website and save it on your CentOS 7 installation (or rlogin). Create a directory for this assignment, and unpack the tar file there. You can do this by executing the command:

```
CentOS> tar xf C01Files.tar
```

The file `PartitionInteger.c` contains a trivial, nonworking implementation of the specified function. You will edit this file to complete the function.

## Implementation and Testing

You can compile the given code by executing the command:

```
CentOS> gcc -o driver -std=c99 -Wall driver.c PartitionInteger.c Generator.o checkAnswer.o
```

You can run the test/grading code by executing the command:

```
CentOS> ./driver <name for test case file> <name for results file>
```

Read the comments in `driver.c` for more information.

`driver` will create a file holding test data, and a file showing the results of comparing your results to correct results, which might be something like this once you've completed your solution:

### Test data

```
even      0
odd       0
even     76266220
odd     188264262
even     959715
odd     319735555
even    167590315
even      7
odd     79336151
even     70727
even    40703067
even     77756
even    98136188
odd     72198246
even    83134010
even    398071022
even    187489318
odd       87
odd     7101
odd     67799567
```

### Test results

```
Correct!           0  even      0
Correct!           0  odd       0
Correct!          76266220  even     62662207
Correct!         188264262  odd     188264262
Correct!          959715  even     959715
Correct!         319735555  odd     319735555
Correct!         167590315  even    601759315
Correct!           7  even      7
Correct!          79336151  odd     79331516
Correct!           70727  even      2777
Correct!         40703067  even    40006737
Correct!          77756  even      67775
Correct!         98136188  even    86889131
Correct!         72198246  odd     71928246
Correct!         83134010  even    84003131
Correct!         398071022  even    800223971
Correct!         187489318  even    848817931
Correct!           87  odd       78
Correct!           7101  odd      7110
Correct!         67799567  odd     77995766
Score: 200 / 200
```

If you try this with the original version of `PartitionInteger.c`, the code will compile, but the results will be (mostly) incorrect. Each execution of `driver` will produce a different set of test data, unless you use the `-repeat` switch:

```
CentOS> ./driver <name of test case file> <name of results file> -repeat
```

In that case, the test data file created by a previous run will be reused. That allows you to focus on a fixed set of test cases while you are debugging.

You should test your solution thoroughly; the given testing code generates random test data, and there is no guarantee that it will cover all cases unless you run it a number of times.

## Submission and Grading

You should not submit your solution to the Curator until you can correctly pass tests with the given testing/grading code.

Submit your completed version of `PartitionInteger.c`, after making changes and testing. Your submission will be compiled, tested and graded by using the supplied code, but that will be done manually after the due date. A TA will also check to see if your solution violates any of the restrictions given in the header comment for the function; if so, your submission will be assigned a score of zero (0), regardless of how many tests it passes.

If you make multiple submissions of your solution to the Curator, we will grade your last submission.

The *Student Guide* and other pertinent information, such as the link to the proper submit page, can be found at:

<http://www.cs.vt.edu/curator/>

## Pledge

Each of your program submissions must be pledged to conform to the Honor Code requirements for this course. Specifically, you **must** include the following pledge statement in the submitted file:

```
// On my honor:
//
// - I have not discussed the C language code in my program with
//   anyone other than my instructor or the teaching assistants
//   assigned to this course.
//
// - I have not used C language code obtained from another student,
//   the Internet, or any other unauthorized source, either modified
//   or unmodified.
//
// - If any C language code or documentation used in my program
//   was obtained from an authorized source, such as a text book or
//   course notes, that has been clearly noted with a proper citation
//   in the comments of my program.
//
// - I have not designed this program in such a way as to defeat or
//   interfere with the normal operation of the Curator System.
//
// <Student Name>
// <Student's VT email PID>
```

**We reserve the option of assigning a score of zero to any submission that is undocumented or does not contain this statement.**