CSE 220: Systems Fundamentals I

Stony Brook University

Programming Assignment #4

Fall 2020

Assignment Due: Sunday, November 8, 2020 by 11:59 pm EST

Updates to this Document

- 10/28/2020: Part 6: confirmed that the `times_sold` field of the new `Book` struct must be set to 0.
- 10/28/2020: Part 7, Example 3 showed the wrong hash table contents; now fixed.
- 10/28/2020: Correction to Part 11 on page 26.
- 10/27/2020: A change to the return value of Part 9 has been made.

Learning Outcomes

After completion of this programming project you should be able to:

- Implement non-trivial algorithms that require conditional execution and iteration.
- Design and implement functions that implement the MIPS assembly function calling conventions.
- Implement algorithms that process 1D arrays of structs of varying size.

Getting Started

Visit the course website and download the files hwk4.zip and MarsFall2020.jar. Fill in the following information at the top of hwk4.asm:

1. your first and last name as they appear in Blackboard
2. your Net ID (e.g., jsmith)
3. your Stony Brook ID # (e.g., 111999999)

Having this information at the top of the file helps us locate your work. If you forget to include this information but don't remember until after the deadline has passed, don't worry about it – we will track down your submission.

Inside hwk4.asm you will find several function stubs that consist simply of `jr $ra` instructions. Your job
in this assignment is to implement all the functions as specified below. Do not change the function names, as the grading scripts will be looking for functions of the given names. However, you may implement additional helper functions of your own, but they must be saved in hwk4.asm. Helper functions will not necessarily be graded, but might be spot-checked to ensure you are following the MIPS function calling conventions.

If you are having difficulty implementing these functions, write out pseudocode or implement the functions in a higher-level language first. Once you understand the algorithm and what steps to perform, then translate the logic into MIPS assembly code.

Be sure to initialize all of your values (e.g., registers) within your functions. Never assume registers or memory will hold any particular values (e.g., zero). MARS initializes all of the registers and bytes of main memory to zeroes. The grading scripts will fill the registers and/or main memory with random values before calling your functions.

Finally, do not define a `.data` section in your hwk4.asm file. A submission that contains a `.data` section will probably result in a score of zero.

Important Information about CSE 220 Programming Projects

- Read the entire project description document twice before starting. Questions posted on Piazza whose answers are clearly stated in the documents will be given lowest priority by the course staff.
- You must use the Stony Brook version of MARS posted on the course website. Do not use the version of MARS posted on the official MARS website. The Stony Brook version has a reduced instruction set, added tools, and additional system calls you might need to complete the homework assignments.
- When writing assembly code, try to stay consistent with your formatting and to comment as much as possible. It is much easier for your TAs and the professor to help you if we can quickly figure out what your code does.
- You personally must implement the programming assignments in MIPS Assembly language by yourself. You may not write or use a code generator or other tools that write any MIPS code for you. You must manually write all MIPS assembly code you submit as part of the assignments.
- Do not copy or share code. Your submissions will be checked against other submissions from this semester and from previous semesters.
- Do not submit a file with the function/label main defined. You are also not permitted to start your label names with two underscores (__). You will obtain a zero for the assignment if you do this.
- Submit your final .asm file to the course website by the due date and time. Late work will be penalized as described in the course syllabus. Code that crashes and cannot be graded will earn no credit. No changes to your submission will be permitted once the deadline has passed.

How Your CSE 220 Assignments Will Be Graded

With few exceptions, all aspects of your programming assignments will be graded entirely through automated means. Grading scripts will execute your code with input values (e.g., command-line arguments, function arguments) and will check for expected results (e.g., print-outs, return values, etc.)

For this homework assignment you will be writing functions in assembly language. The functions will be tested independently of each other. This is very important to note, as you must take care that no function you write ever has side-effects or requires that other functions be called before the function in question is called. Both of these are generally considered bad practice in programming.

Some other items you should be aware of:

- Each test case must execute in 1,000,000 instructions or fewer. Efficiency is an important aspect of programming. This maximum instruction count will be increased in cases where a complicated algorithm might be necessary, or a large data structure must be traversed. To find the instruction count of your code in MARS, go to the **Tools** menu and select **Instruction Statistics**. Press the button marked **Connect to MIPS**. Then assemble and run your code as normal.

- Any excess output from your program (debugging notes, etc.) will impact grading. Do not leave erroneous print-outs in your code.

- We will provide you with a small set of test cases for each assignment to give you a sense of how your work will be graded. It is your responsibility to test your code thoroughly by creating your own test cases.

- The testing framework we use for grading your work will not be released, but the test cases and expected results used for testing will be released.

Register Conventions

You must follow the register conventions taught in lecture and reviewed in recitation. Failure to follow them will result in loss of credit when we grade your work. Here is a brief summary of the register conventions and how your use of them will impact grading:

- It is the callee's responsibility to save any $s registers it overwrites by saving copies of those registers on the stack and restoring them before returning.

- If a function calls a secondary function, the caller must save $ra before calling the callee. In addition, if the caller wants a particular $a, $t or $v register's value to be preserved across the secondary function call, the best practice would be to place a copy of that register in an $s register before making the function call.

- A function which allocates stack space by adjusting $sp must restore $sp to its original value before returning.

- Registers $fp and $gp are treated as preserved registers for the purposes of this course. If a function modifies one or both, the function must restore them before returning to the caller. There really is no reason for your code to touch the $gp register, so leave it alone.

The following practices will result in loss of credit:

- “Brute-force” saving of all $s registers in a function or otherwise saving $s registers that are not overwritten by a function.
- Callee-saving of $a, $t or $v registers as a means of “helping” the caller.
- “Hiding” values in the $k, $f and $at registers or storing values in main memory by way of offsets to $gp. This is basically cheating or at best, a form of laziness, so don’t do it. We will comment out any such code we find.

How to Test Your Functions

To test your implemented functions, open the provided “main” files in MARS. Next, assemble the main file and run it. MARS will include the contents of any .asm files referenced with the .include directive(s) at the end of the file and then append the contents of your hwk4.asm file before assembling the program.

Each main file calls a single function with one of the sample test cases and prints any return value(s). You will need to change the arguments passed to the functions to test your functions with the other cases. To test each of your functions thoroughly, create your own test cases in those main files. Your submission will not be graded using the examples provided in this document or using the provided main file(s). Do not submit your main files with your hwk4.asm file – we will delete them. Also please note that these testing main files will not be used in grading.

Again, any modifications to the main files will not be graded. You will submit only your hwk4.asm for grading. Make sure that all code required for implementing your functions is included in the hwk4.asm file. To make sure that your code is self-contained, try assembling your hwk4.asm file by itself in MARS. If you get any errors (such as a missing label), this means that your hwk4.asm file is attempting to reference labels in the main file, which will not have the same names during grading!

A Final Reminder on How Your Work Will be Graded

It is imperative (crucial, essential, necessary, critically important) that you implement the functions below exactly as specified. Do not deviate from the specifications, even if you think you are implementing the program in a better way. Modify the contents of memory only as described in the function specifications!

Preliminaries

For this assignment you will be implementing a hash table data structure for storing different types of structs. The context of the assignment is a simulated book store that will track its books for sale and the actual sales themselves through two different hash tables. Towards the end of the assignment we will also explore a (sorta related) interesting computational problem we will solve using a simple, non-recursive brute-force algorithm. Perhaps in your Algorithms course you will study it (or a similar problem) using a more efficient, recursive algorithm.

Data Structures

The assignment relies on three data structures given as structs:

```
struct Book {
   string isbn;      // 13-character, null-terminated string [14 bytes]
   string title;     // 25-byte buffer to store the null-terminated title
   string author;    // 25-byte buffer to store the null-terminated author
   int times_sold;   // 4-byte integer the stores # of sales of this book
}
```

Note that a `Book` struct requires 14 + 25 + 25 + 4 = 68 bytes. We will assume that the entire struct is word-aligned, which guarantees that `times_sold` will also be word-aligned. In this assignment we will also assume that a string containing an ISBN consists only of the digit characters (no dashes, spaces, X's, etc.)

```
struct BookSale {
   string isbn;       // 13-character, null-terminated string [14 bytes]
   byte[2] padding;   // 2 bytes of zeros
   int customer_id;   // 4-byte customer ID#
   int sale_date;     // date of sale as # of days since 1/1/1600 [4 bytes]
   int sale_price;    // 4 bytes
}
```

Note that a `BookSale` struct requires 14 + 2 + 4 + 4 + 4 = 28 bytes. The date of the sale is encoded as a 4-byte integer that provides the number of days that have passed since January 1, 1600. For example, January 5, 1600 would be represented by the number 4. January 1, 1600 itself would be represented by the number 0. November 8, 2020 would be represented by the number 153,714. You will write functions to convert a date-string in the form YYYY-MM-DD (e.g., "2020-11-08") to this numerical encoding. The inspiration for this time format is the UNIX timestamp system.

Instances of the above structs will be stored in two separate hash table data structures, as defined below. The hash table itself will be word-aligned.

```
struct Hashtable {
   int capacity;        // maximum # of elements in hashtable [4 bytes]
   int size;            // # of elements in hashtable [4 bytes]
   int element_size;    // size of one element in the hashtable [4 bytes]
   object[] elements;   // objects stored in the hashtable
                        // consumes (capacity*element_size) bytes
}
```

A hash table in this assignment consists of an array of structs, **not** an array of pointers to structs. Note how this differs from Java, wherein what we call "an array of objects" is really an array of *references* to objects. In C and MIPS it *is* possible to create an array of pointers to objects, and we will explore that option in the last function of the assignment. But for the functions in the current assignment, we will store the structs directly in the arrays themselves, the reason being that one of the purposes of this assignment is to deal with arrays of varying element sizes.

In the `elements` array, an *empty* entry, i.e., an entry in the array that has never stored a struct, is filled entirely with zeros. A *deleted* (sometimes called *available*) entry is one that once stored an object that has since been deleted (removed) from the hash table. A deleted element is represented by `(capacity*element_size)`, one-byte, two's complement representations of -1 (i.e., 0xFF). That is, the entire entry in the array consists of one-byte -1s. Due to a quirk of how we will be using the hash table for this assignment, it will be sufficient to load the first byte of an entry in the `elements` array and check if that byte equals 0xFF. If so, that entry can be assumed to be *deleted*. Similarly, to check if an entry in `elements` is *empty*, it will be sufficient to load the first byte and check if it equals zero. Every 13-character ISBN must start with '9', so the first character of a non-empty, non-deleted object in `elements` cannot be' '0'. In sum, the first byte of a `Book` struct or a `BookSale` struct must be the number 0, the number -1, or the ASCII code for the character '9'.

Part 1: Copy a Memory Buffer

```
int memcpy(byte[] dest, byte[] src, int n)
```

The `memcpy` function copies `n` bytes of data from address `src` to address `dest`. You may assume that the `dest` buffer is at least `n` bytes in size. This function will be used in later functions to copy structs from one place in memory to another. Therefore, you should code for generality and not be concerned about the significance of the bytes that the function copies. For this reason, your code should use `lbu` to read the bytes from memory.

The function takes the following arguments, in this order:

- `dest`: the address to copy bytes to
- `src`: the address to copy bytes from
- `n`: the number of bytes to copy (must be greater than 0)

Note that `src` and `dest` are not necessarily word-aligned.

Returns in $v0:

- `n` if `n` > 0, or
- -1 if `n` ≤ 0

Additional requirements:

- The function may only change the contents of the `dest` buffer and no other parts of main memory.
- Only the first `n` bytes memory starting at `dest` may be changed. All other bytes in `dest` must be left unchanged.

Examples:

Function Arguments	Return Value	Updated `dest`
`"XXXXXXX", "ABCDEFGHIJ", 3`	`3`	`"ABCXXXX"`
`"XXXXXXX", "ABCDEFGHIJ", 7`	`7`	`"ABCDEFG"`
`"XXXXXXX", "ABCDEFGHIJ", -3`	`-1`	`"XXXXXXX"`
`"XXXXXXX", "ABCDEFGHIJ", 0`	`-1`	`"XXXXXXX"`

Part 2: Compare Two Strings

```
int strcmp(string str1, string str2)
```

The `strcmp` function takes two null-terminated strings (either or both of which could be empty) and returns an integer that indicates the lexicographic order of the strings. The function begins by comparing the first character of each string. If they are equal to each other, it continues with the following pairs until the characters differ or until a terminating null-character is reached. Note that the strings can be of different lengths.

The function takes the following arguments, in this order:

- `str1`: the starting address of the first string
- `str2`: the starting address of the second string

Returns in $v0:

- the difference between the ASCII values of the first mismatch, if any: `str1[n] - str2[n]`, where `n` is the index of the first mismatch; or
- 0 if the contents of both strings are identical (including the case where they are both empty strings); or
- length of `str1` if `str2` is an empty string but `str1` is non-empty; or
- negated length of `str2` if `str1` is an empty string but `str2` is non-empty

Additional requirements:

- The function must not write any changes to main memory.

Examples:

Function Arguments	Return Value
`"ABCD", "ABCGG"`	`-3`
`"WHOOP!", "WHOA"`	`14`
`"Intel", "pentium"`	`-39`
`"STONY", "BROOK"`	`17`

`"", "mouse"`	`-5`
`"lonely guy", ""`	`10`
`"Wolfie", "Wolfie"`	`0`
`"", ""`	`0`
`"happy", "Z"`	`14`
`"WOLF", "WOLFIE"`	`-73`
`"StonyBrook", "Stony"`	`66`

Part 3: Initialize a Hash Table

```
int initialize_hashtable(Hashtable* hashtable, int capacity,
                              int element_size)
```

The `initialize_hashtable` function takes a pointer to (i.e., address of) an uninitialized `Hashtable` struct and performs the following steps:

1. sets the struct's `capacity` field to the `capacity` argument
2. sets the struct's `size` field to 0
3. sets the struct `element_size` field to the `element_size` argument
4. fills the entirety of the struct's `elements` field to 0 (i.e., all `capacity * element_size` bytes)

Assume that the uninitialized block of memory referenced by `hashtable` is large enough to store a hash table of the required size in bytes.

The function takes the following arguments, in this order:

- `hashtable`: a pointer to an uninitialized block of memory
- `capacity`: the number of elements in the hash table's `elements` array
- `element_size`: the size of a single struct stored in one element of the hash table's `elements` array.

Returns in $v0:

- -1 if `capacity` is less than 1 or `element_size` is less than 1; or
- 0 otherwise

Additional requirements:

- The function may only change the contents of the `hashtable` buffer and no other parts of memory.

Example #1: initializing a hash table to store `Book` structs.

`hashtable` = pointer to buffer of 352 (12+5*68) uninitialized bytes of memory

```
capacity = 5
element_size = 68
```

Return value in $v0: 0

Resulting contents of `hashtable`:

```
5 0 68 (capacity, size, element_size)
```

`elements[]` array contents in hexadecimal, truncated:

```
0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
1: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
2: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
3: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
4: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
```

Example #2: initializing a hash table to store `BookSale` structs.

`hashtable` = pointer to buffer of 320 (12+11*28) uninitialized bytes of memory

```
capacity = 11
element_size = 28
```

Return value in $v0: 0

Resulting contents of `hashtable`:

```
11 0 28 (capacity, size, element_size)
```

`elements[]` array contents in hexadecimal, truncated:

```
0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
1: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
2: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
3: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
4: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
.
.
.
10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 …
```

Example #3: failed attempt to initialize a hash table to store `BookSale` structs.

`hashtable` = pointer to buffer of some unknown number of uninitialized bytes of memory

```
capacity = -3
element_size = 28
```

Return value in $v0: -1

Contents of `hashtable` are the same as before the function call.

Part 4: Compute the Hash Code for an ISBN in a Given Hash Table

```
int hash_book(Hashtable* books, string isbn)
```

The `hash_book` function takes a pointer to a valid `Hashtable` struct named `books` and an ISBN-13 value called `isbn` for which we want to compute the hash function. The hash function itself is very simple: sum the ASCII codes of the 13 characters in `isbn` and take the modulus of dividing that sum by the capacity of the `books` hash table (e.g., sum of the ASCII codes *mod* the capacity).

The function takes the following arguments, in this order:

- `books`: the starting address of a valid `Hashtable` struct
- `isbn`: a 13-character, null-terminated string that represents a valid ISBN

Returns in $v0:

- The value of the hash function when evaluated for the provided ISBN.

Additional requirements:

- The function may not write any changes to main memory.

Example #1:

`hashtable` = pointer to a valid hash table with capacity = 9
`isbn = "9780671028370"`
Return value in $v0: 7

Example #2:

`hashtable` = pointer to a valid hash table with capacity = 17
`isbn = "9780071401940"`
Return value in $v0: 11

Example #3:

`hashtable` = pointer to a valid hash table with capacity = 13
`isbn = "1123121401940"`
Return value in $v0: 3

Part 5: Retrieve a Book from a Hashtable of `Book` Structs

```
int, int get_book(Hashtable* books, string isbn) -> (int, int)
```

The `get_book` function attempts to find a struct inside `books` with the given ISBN and returns the index of that struct, as well as the number of entries in `books.elements` that needed to be inspected to find that struct by way of linear probing. If an *empty* entry is encountered during probing, this means that no book of the given ISBN is in the hash table. If a *deleted* entry is encountered during probing, the linear probing must continue. If no struct has the given ISBN, the function returns -1 in $v0 and the number of inspected entries of elements in $v1.

The function takes the following arguments, in this order:

- `books`: the starting address of a valid `Hashtable` struct
- `isbn`: a 13-character null-terminated string that represents a valid ISBN

Returns in $v0:

- The index in `elements` where the book was found; or
- -1 if the hash table does not contain a book with the given ISBN

Returns in $v1:

- The number of entries in the `elements` array that were accessed before a book of the given ISBN was found or was determined not to be in the `elements` array.

Additional requirements:

- The function may not write any changes to main memory.
- The function must call `hash_book` and `strcmp`.

The examples in this document are not comprehensive nor cover all possible classes of inputs!

Example #1: Attempt to get a book from an empty hash table

```
isbn = "9780553214830"
```
Contents of `books` hash table:
```
9 0 68 (capacity, size, element_size)
0: empty
1: empty
2: empty
3: empty
4: empty
5: empty
6: empty
7: empty
8: empty
```
Return value in $v0: -1
Return value in $v1: 1

Example #2: Get a book that is present in the hash table at the expected index

```
isbn = "9780440060670"
```
Contents of `books` hash table:
```
7 5 68 (capacity, size, element_size)
0: 9780345501330\0Fairy Tail, Vol. 1 (Fair\0Hiro Mashima, William Fl\00
1: empty
2: 9780670032080\0Financial Peace Revisite\0Dave
                              Ramsey\0\0\0\0\0\0\0\0\0\0\0\0\0\00
3: 9780440060670\0The Other Side of Midnig\0Sidney
```

```
                           Sheldon\0\0\0\0\0\0\0\0\0\0\00
4: 9780064408330\0Joey Pigza Swallowed the\0Jack
                           Gantos\0\0\0\0\0\0\0\0\0\0\0\0\0\00
5: deleted
6: 9781416971700\0Out of My Mind\0\0\0\0\0\0\0\0\0\0\0Sharon M.
                           Draper\0\0\0\0\0\0\0\0\0\00
```

Return value in $v0: 3
Return value in $v1: 1

Example #3: Attempt to get a book that is not present in the hash table; expected location is empty

```
isbn = "9780007201780"
```

Contents of `books` hash table:

```
7 5 68 (capacity, size, element_size)
0: 9780345501330\0Fairy Tail, Vol. 1 (Fair\0Hiro Mashima, William Fl\00
1: empty
2: 9780670032080\0Financial Peace Revisite\0Dave
                           Ramsey\0\0\0\0\0\0\0\0\0\0\0\0\0\00
3: 9780440060670\0The Other Side of Midnig\0Sidney
                           Sheldon\0\0\0\0\0\0\0\0\0\0\00
4: 9780064408330\0Joey Pigza Swallowed the\0Jack
                           Gantos\0\0\0\0\0\0\0\0\0\0\0\0\0\00
5: deleted
6: 9781416971700\0Out of My Mind\0\0\0\0\0\0\0\0\0\0\0Sharon M.
                           Draper\0\0\0\0\0\0\0\0\0\00
```

Return value in $v0: -1
Return value in $v1: 1

Example #4: Get a book that is not present in the hash table; expected location is deleted (hash table not full); probing required

```
isbn = "9780060182980"
```

Contents of `books` hash table:

```
7 5 68 (capacity, size, element_size)
0: 9780345501330\0Fairy Tail, Vol. 1 (Fair\0Hiro Mashima, William Fl\00
1: empty
2: 9780670032080\0Financial Peace Revisite\0Dave
                           Ramsey\0\0\0\0\0\0\0\0\0\0\0\0\0\00
3: 9780440060670\0The Other Side of Midnig\0Sidney
                           Sheldon\0\0\0\0\0\0\0\0\0\0\00
4: 9780064408330\0Joey Pigza Swallowed the\0Jack
                           Gantos\0\0\0\0\0\0\0\0\0\0\0\0\0\00
5: deleted
6: 9781416971700\0Out of My Mind\0\0\0\0\0\0\0\0\0\0\0Sharon M.
                           Draper\0\0\0\0\0\0\0\0\0\00
```

Return value in $v0: -1
Return value in $v1: 6

Example #5: Get a book that is present in a full hash table

```
isbn = "9780064408330"
```
Contents of `books` hash table:
```
7 7 68 (capacity, size, element_size)
0: 9780345501330\0Fairy Tail, Vol. 1 (Fair\0Hiro Mashima, William Fl\00
1: 9780345419580\0Moreta: Dragonlady of Pe\0Anne
                           McCaffrey\0\0\0\0\0\0\0\0\0\00
2: 9780670032080\0Financial Peace Revisite\0Dave
                           Ramsey\0\0\0\0\0\0\0\0\0\0\0\0\00
3: 9780440060670\0The Other Side of Midnig\0Sidney
                           Sheldon\0\0\0\0\0\0\0\0\0\0\00
4: 9780064408330\0Joey Pigza Swallowed the\0Jack
                           Gantos\0\0\0\0\0\0\0\0\0\0\0\0\00
5: 9780140168130\0Big Sur\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0Jack Kerouac,
                           Aram Saroy\00
6: 9781416971700\0Out of My Mind\0\0\0\0\0\0\0\0\0\0\0Sharon M.
Draper\0\0\0\0\0\0\0\0\00
```
Return value in $v0: 4
Return value in $v1: 1

Example #6: Get a book that is present in the hash table at a probed index; one or more deleted slots encountered during probing

```
isbn = "9780140168130"
```
Contents of `books` hash table:
```
7 5 68 (capacity, size, element_size)
0: 9780345501330\0Fairy Tail, Vol. 1 (Fair\0Hiro Mashima, William Fl\00
1: 9780345419580\0Moreta: Dragonlady of Pe\0Anne
                           McCaffrey\0\0\0\0\0\0\0\0\0\00
2: deleted
3: 9780440060670\0The Other Side of Midnig\0Sidney
                           Sheldon\0\0\0\0\0\0\0\0\0\0\00
4: deleted
5: 9780140168130\0Big Sur\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0Jack Kerouac,
                           Aram Saroy\00
6: 9781416971700\0Out of My Mind\0\0\0\0\0\0\0\0\0\0\0Sharon M.
                           Draper\0\0\0\0\0\0\0\0\00
```
Return value in $v0: 5
Return value in $v1: 6

Example #7: Attempt to get a book that is not present in the hash table; hash table is full

```
isbn = "9780316905750"
```
Contents of `books` hash table:
```
7 7 68 (capacity, size, element_size)
0: 9780345501330\0Fairy Tail, Vol. 1 (Fair\0Hiro Mashima, William Fl\00
1: 9780345419580\0Moreta: Dragonlady of Pe\0Anne
                          McCaffrey\0\0\0\0\0\0\0\0\0\0\00
2: 9781516865870\0Beacon 23: The Complete \0Hugh
                          Howey\0\0\0\0\0\0\0\0\0\0\0\0\0\00
3: 9780440060670\0The Other Side of Midnig\0Sidney
                           Sheldon\0\0\0\0\0\0\0\0\0\0\00
4: 9781573451990\0Rumors of War (Children \0Dean
                           Hughes\0\0\0\0\0\0\0\0\0\0\0\0\00
5: 9780140168130\0Big Sur\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0Jack Kerouac,
                           Aram Saroy\00
6: 9781416971700\0Out of My Mind\0\0\0\0\0\0\0\0\0\0\0Sharon M.
                           Draper\0\0\0\0\0\0\0\0\0\00
```
Return value in $v0: -1
Return value in $v1: 7

Part 6: Insert a `Book` Struct into a Hashtable of Books

```
int, int add_book(Hashtable* books, string isbn, string title,
                  string author)
```

The `add_book` function attempts to insert a new `Book` struct into a given hash table by initializing the contents of the struct with the provided ISBN, title and author. It also sets the `times_sold` field of the struct to 0. It begins by checking if the hash table is full. If so, the function returns (-1, -1) and makes no changes to the hash table's contents. Next, it checks if a book of the given ISBN is already in the hash table. If so, the function returns the return values of `get_book(books, isbn)` as its own return values. Otherwise, the function attempts to insert the struct at `books.elements[hash_book(isbn)]`. Failing that, it performs linear probing to find the next *empty* or *deleted* entry in the `elements` array and inserts the struct at that location. If needed, probing wraps-around from the last entry in `elements` to the first. The function returns in $v0 the index where the struct was inserted. It returns in $v1 the number of entries in `elements` it had to read before finding an empty or deleted entry. For example, if it must skip over 3 occupied slots before finding an empty one, $v1 would be 4 in that case. If a book is successfully added to the hash table, `books.size` is incremented by 1.

Note that at most 24 characters of a book's title or author can be stored in a `Book` struct. Therefore, any characters in `title` or `author` beyond the 24th character are dropped. The function must then write a null-terminator for the truncated string.

When writing the `title` and `author` string arguments into the `Book` struct, if the string is shorter than 24 characters in length, the function pads out the remainder of the 25 bytes of the `title` or `author` field by appending null-terminators. For instance, if `title` were `"MIPS Rocks"`, which is only 10 characters in length, the function would write the following as the `title` field in the new `Book` struct:

```
MIPS Rocks\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
```

The function takes the following arguments, in this order:

- `books`: the starting address of a valid `Hashtable` of `Book` structs
- `isbn`: a 13-character null-terminated string that represents a valid ISBN
- `title`: a non-empty, null-terminated string that represents a book's title
- `author`: a non-empty, null-terminated string that represents a book's author

Returns in $v0:

- The index in `elements` where the new book was added (or found, if the book was already in the hash table); or
- -1 if the hash table was full before the function call.

Returns in $v1:

- The number of entries in the `elements` array that were accessed before the book was added to the hash table; or
- -1 if the hash table was full before the function call.

Additional requirements:

- The function may only change the contents of the `books` hash table and no other parts of memory.
- The function must call `get_book`, `hash_book` and `memcpy`.

Example #1: Add a book to an empty hash table.

```
isbn = "9780553214830"
title = "The Declaration of Independence"
author = "Founding Fathers"
```

Return value in $v0: 4
Return value in $v1: 1
Resulting `books` contents:

```
9 1 68 (capacity, size, element_size)
0: empty
1: empty
2: empty
3: empty
4: 9780553214830\0The Declaration of Indep\0Founding
                                                Fathers\0\0\0\0\0\0\0\0\00
5: empty
6: empty
```

```
7: empty
8: empty
```

(Note: null-terminators are appended as needed to pad the `title` and `author` fields to 25 bytes each. The final 4-byte word in each struct is the `times_sold` field, which is 0 in this case.)

Example #2: Add a book to a hash table with existing values; no collisions

isbn = "9781620610090"
title = "Opal (Lux, #3)"
author = "Jennifer L. Armentrout"
`books` contents before function call:

```
9 4 68 (capacity, size, element_size)
0: empty
1: 9780446695660\0Double Whammy\0\0\0\0\0\0\0\0\0\0\0Carl
Hiaasen\0\0\0\0\0\0\0\0\0\0\0\0\00
2: empty
3: empty
4: 9780553214830\0The Declaration of Indep\0Founding
                                             Fathers\0\0\0\0\0\0\0\0\014
5: 9788433914260\0Hollywood\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0Charles
                                             Bukowski\0\0\0\0\0\0\0\0\00
6: 9780345527780\0Wicked Business (Lizzy &\0Janet

Evanovich\0\0\0\0\0\0\0\0\0\00
7: empty
8: empty
```

Return value in $v0: 7
Return value in $v1: 1
Resulting `books` contents:

```
9 5 68 (capacity, size, element_size)
0: empty
1: 9780446695660\0Double Whammy\0\0\0\0\0\0\0\0\0\0\0Carl
                           Hiaasen\0\0\0\0\0\0\0\0\0\0\0\0\00
2: empty
3: empty
4: 9780553214830\0The Declaration of Indep\0Founding
                           Fathers\0\0\0\0\0\0\0\0\014
5: 9788433914260\0Hollywood\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0Charles
                           Bukowski\0\0\0\0\0\0\0\0\00
6: 9780345527780\0Wicked Business (Lizzy &\0Janet
                           Evanovich\0\0\0\0\0\0\0\0\0\00
7: 9781620610090\0Opal (Lux, #3)\0\0\0\0\0\0\0\0\0\0Jennifer L.
                           Armentrout\0\0\00
8: empty
```

Note that the Declaration of Independence was sold 14 times.

Example #3: Add a book to a hash table with existing values; collision with one probe required

```
isbn = "9781416971700"
title = "Out of My Mind"
author = "Sharon M. Draper"
```
`books` contents before function call:
```
7 3 68 (capacity, size, element_size)
0: 9780345501330\0Fairy Tail, Vol. 1 (Fair\0Hiro Mashima, William Fl\00
1: empty
2: empty
3: empty
4: 9780064408330\0Joey Pigza Swallowed the\0Jack
                           Gantos\0\0\0\0\0\0\0\0\0\0\0\0\00
5: 9780312577220\0Fly Away (Firefly Lane, \0Kristin
                           Hannah\0\0\0\0\0\0\0\0\0\0\00
6: empty
```
Return value in $v0: 6
Return value in $v1: 2
Resulting `books` contents:
```
7 4 68 (capacity, size, element_size)
0: 9780345501330\0Fairy Tail, Vol. 1 (Fair\0Hiro Mashima, William Fl\00
1: empty
2: empty
3: empty
4: 9780064408330\0Joey Pigza Swallowed the\0Jack
                           Gantos\0\0\0\0\0\0\0\0\0\0\0\0\00
5: 9780312577220\0Fly Away (Firefly Lane, \0Kristin
                           Hannah\0\0\0\0\0\0\0\0\0\0\00
6: 9781416971700\0Out of My Mind\0\0\0\0\0\0\0\0\0\0\0Sharon M.
                           Draper\0\0\0\0\0\0\0\0\00
```

Example #4: Add a book to a hash table with existing values; collisions with multiple probes required

```
isbn = "9781455525930"
title = "Crimson Shore (Pendergast, #15)"
author = "Douglas Preston, Lincoln Child"
```
`books` contents before function call:
```
11 7 68 (capacity, size, element_size)
0: empty
1: 9780345501330\0Fairy Tail, Vol. 1 (Fair\0Hiro Mashima, William Fl\0251
2: 9781416971700\0Out of My Mind\0\0\0\0\0\0\0\0\0\0\0Sharon M.
                           Draper\0\0\0\0\0\0\0\0\00
```

```
3: 9780156948780\0The Waste Land and Other\0T.S.
                           Eliot\0\0\0\0\0\0\0\0\0\0\0\0\0\00
4: 9780670032080\0Financial Peace Revisite\0Dave
                           Ramsey\0\0\0\0\0\0\0\0\0\0\0\0\0422
5: 9780064408330\0Joey Pigza Swallowed the\0Jack
                           Gantos\0\0\0\0\0\0\0\0\0\0\0\0\01251
6: 9780312577220\0Fly Away (Firefly Lane, \0Kristin
                           Hannah\0\0\0\0\0\0\0\0\0\0\0734
7: empty
8: empty
9: empty
10: 9780060855900\0Equal Rites (Discworld, \0Terry
                           Pratchett\0\0\0\0\0\0\0\0\0\00
```

Return value in $v0: 7

Return value in $v1: 3

Resulting `books` contents:

```
11 8 68 (capacity, size, element_size)
0: empty
1: 9780345501330\0Fairy Tail, Vol. 1 (Fair\0Hiro Mashima, William Fl\0251
2: 9781416971700\0Out of My Mind\0\0\0\0\0\0\0\0\0\0Sharon M.
                           Draper\0\0\0\0\0\0\0\0\00
3: 9780156948780\0The Waste Land and Other\0T.S.
                           Eliot\0\0\0\0\0\0\0\0\0\0\0\0\0\00
4: 9780670032080\0Financial Peace Revisite\0Dave
                           Ramsey\0\0\0\0\0\0\0\0\0\0\0\0\0422
5: 9780064408330\0Joey Pigza Swallowed the\0Jack
                           Gantos\0\0\0\0\0\0\0\0\0\0\0\0\01251
6: 9780312577220\0Fly Away (Firefly Lane, \0Kristin
                           Hannah\0\0\0\0\0\0\0\0\0\0\0734
7: 9781455525930\0Crimson Shore (Pendergas\0Douglas Preston, Lincoln\00
8: empty
9: empty
10: 97800608550\0Equal Rites (Discworld, \0Terry
                           Pratchett\0\0\0\0\0\0\0\0\0\00
```

Note how the title and author were truncated when stored in the hash table.

Example #5: Add a book to a hash table with existing values; collisions with multiple probes required

```
isbn = "9780451230230"
title = "The Pacific"
author = "Hugh Ambrose"
```

`books` contents before function call:

```
7 4 68 (capacity, size, element_size)
0: 9780345501330\0Fairy Tail, Vol. 1 (Fair\0Hiro Mashima, William Fl\07777
1: 0
2: -1
3: 9780670032080\0Financial Peace Revisite\0Dave
                           Ramsey\0\0\0\0\0\0\0\0\0\0\0\0\0222
4: 9780064408330\0Joey Pigza Swallowed the\0Jack
                           Gantos\0\0\0\0\0\0\0\0\0\0\0\0\0333
5: -1
6: 9781416971700\0Out of My Mind\0\0\0\0\0\0\0\0\0\0Sharon M.
Draper\0\0\0\0\0\0\0\0\00
```

Return value in $v0: 5

Return value in $v1: 3

Resulting `books` contents:

```
7 5 68 (capacity, size, element_size)
0: 9780345501330\0Fairy Tail, Vol. 1 (Fair\0Hiro Mashima, William Fl\07777
1: 0
2: -1
3: 9780670032080\0Financial Peace Revisite\0Dave
                           Ramsey\0\0\0\0\0\0\0\0\0\0\0\0\0222
4: 9780064408330\0Joey Pigza Swallowed the\0Jack
                           Gantos\0\0\0\0\0\0\0\0\0\0\0\0\0333
5: 9780451230230\0The Pacific\0\0\0\0\0\0\0\0\0\0\0\0\0\0Hugh
                           Ambrose\0\0\0\0\0\0\0\0\0\0\0\0\00
6: 9781416971700\0Out of My Mind\0\0\0\0\0\0\0\0\0\0Sharon M.
                           Draper\0\0\0\0\0\0\0\0\00
```

Part 7: Delete a `Book` Struct from a Hash Table of Books

`int delete_book(Hashtable* books, string isbn)`

The `delete_book` function attempts to delete the `Book` struct from the hash table with the given ISBN. If the book exists, then its entire struct is filled with the value -1 (8-bit, two's complement format) and the function returns the index in books.elements where the book was found. If a book is successfully deleted from the hash table, `books.size` is decremented by 1. If the book is not found, the function returns -1 and makes no changes to the contents of `books`.

The function takes the following arguments, in this order:

- `books`: the starting address of a valid `Hashtable` struct
- `isbn`: a 13-character null-terminated string that represents a valid ISBN

Returns in $v0:

- The index where the deleted book had been located; or
- -1 if no book with the given ISBN was found in the hash table

Additional requirements:

- The function may only change the contents of the `books` hash table and no other parts of memory.
- The function must call `get_book`.

Example #1: Attempt to delete a book from an empty hash table

```
isbn = "9780553214830"
```
`books` contents before function call:
```
7 0 68 (capacity, size, element_size)
0: empty
1: empty
2: empty
3: empty
4: empty
5: empty
6: empty
```
Return value in $v0: -1
Resulting `books` contents:
```
7 0 68 (capacity, size, element_size)
0: empty
1: empty
2: empty
3: empty
4: empty
5: empty
6: empty
```

Example #2: Delete a book that is present in the hash table at the expected index

```
isbn = "9780060855900"
```
`books` contents before function call:
```
7 5 68 (capacity, size, element_size)
0: 9780345501330\0Fairy Tail, Vol. 1 (Fair\0Hiro Mashima, William Fl\00
1: empty
2: 9780060855900\0Equal Rites (Discworld, \0Terry
                          Pratchett\0\0\0\0\0\0\0\0\0\00
3: 9780345527780\0Wicked Business (Lizzy &\0Janet
                          Evanovich\0\0\0\0\0\0\0\0\0\00
4: 9780064408330\0Joey Pigza Swallowed the\0Jack
                          Gantos\0\0\0\0\0\0\0\0\0\0\0\0\00
5: 9780312577220\0Fly Away (Firefly Lane, \0Kristin
                          Hannah\0\0\0\0\0\0\0\0\0\0\00
6: empty
```
Return value in $v0: 2
Resulting `books` contents:

```
7 4 68 (capacity, size, element_size)
0: 9780345501330\0Fairy Tail, Vol. 1 (Fair\0Hiro Mashima, William Fl\00
1: empty
2: deleted
3: 9780345527780\0Wicked Business (Lizzy &\0Janet
                          Evanovich\0\0\0\0\0\0\0\0\0\00
4: 9780064408330\0Joey Pigza Swallowed the\0Jack
                          Gantos\0\0\0\0\0\0\0\0\0\0\0\0\0\00
5: 9780312577220\0Fly Away (Firefly Lane, \0Kristin
                          Hannah\0\0\0\0\0\0\0\0\0\0\0\00
6: empty
```

Example #3: Attempt to delete a book that is not present in the hash table; expected location is empty

`isbn = "9781439164630"`
`books` contents before function call:

```
7 5 68 (capacity, size, element_size)
0: 9780345501330\0Fairy Tail, Vol. 1 (Fair\0Hiro Mashima, William Fl\00
1: empty
2: 9780060855900\0Equal Rites (Discworld, \0Terry
                          Pratchett\0\0\0\0\0\0\0\0\0\00
3: 9780345527780\0Wicked Business (Lizzy &\0Janet
                          Evanovich\0\0\0\0\0\0\0\0\0\00
4: 9780064408330\0Joey Pigza Swallowed the\0Jack
                          Gantos\0\0\0\0\0\0\0\0\0\0\0\0\0\00
5: 9780312577220\0Fly Away (Firefly Lane, \0Kristin
                          Hannah\0\0\0\0\0\0\0\0\0\0\0\00
6: empty
```

Return value in $v0: -1
Resulting `books` contents:

```
7 5 68 (capacity, size, element_size)
0: 9780345501330\0Fairy Tail, Vol. 1 (Fair\0Hiro Mashima, William Fl\00
1: empty
2: 9780060855900\0Equal Rites (Discworld, \0Terry
                          Pratchett\0\0\0\0\0\0\0\0\0\00
3: 9780345527780\0Wicked Business (Lizzy &\0Janet
                          Evanovich\0\0\0\0\0\0\0\0\0\00
4: 9780064408330\0Joey Pigza Swallowed the\0Jack
                          Gantos\0\0\0\0\0\0\0\0\0\0\0\0\0\00
5: 9780312577220\0Fly Away (Firefly Lane, \0Kristin
                          Hannah\0\0\0\0\0\0\0\0\0\0\0\00
6: empty
```

Example #4: Delete a book that is not present in the hash table; expected location is deleted

```
isbn = "9780064410150"
```
`books` contents before function call:
```
7 3 68 (capacity, size, element_size)
0: empty
1: deleted
2: 9780060855900\0Equal Rites (Discworld, \0Terry
                           Pratchett\0\0\0\0\0\0\0\0\0\00
3: 9780345527780\0Wicked Business (Lizzy &\0Janet
                           Evanovich\0\0\0\0\0\0\0\0\0\00
4: deleted
5: 9780312577220\0Fly Away (Firefly Lane, \0Kristin
                           Hannah\0\0\0\0\0\0\0\0\0\0\00
6: empty
```
Return value in $v0: -1

Resulting `books` contents:
```
7 3 68 (capacity, size, element_size)
0: empty
1: deleted
2: 9780060855900\0Equal Rites (Discworld, \0Terry
                           Pratchett\0\0\0\0\0\0\0\0\0\00
3: 9780345527780\0Wicked Business (Lizzy &\0Janet
                           Evanovich\0\0\0\0\0\0\0\0\0\00
4: deleted
5: 9780312577220\0Fly Away (Firefly Lane, \0Kristin
                           Hannah\0\0\0\0\0\0\0\0\0\0\00
6: empty
```

Example #5: Attempt to delete a book not present in the hash table; hash table is full

```
isbn = "9780802122550"
```
`books` contents before function call:
```
7 7 68 (capacity, size, element_size)
0: 9780553214830\0The Declaration of Indep\0Founding
                           Fathers\0\0\0\0\0\0\0\0\0\00
1: 9780446695660\0Double Whammy\0\0\0\0\0\0\0\0\0\0\0Carl
                           Hiaasen\0\0\0\0\0\0\0\0\0\0\0\0\00
2: 9780060855900\0Equal Rites (Discworld, \0Terry
                           Pratchett\0\0\0\0\0\0\0\0\0\00
3: 9780345527780\0Wicked Business (Lizzy &\0Janet
                           Evanovich\0\0\0\0\0\0\0\0\0\00
4: 9788433914260\0Hollywood\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0Charles
                           Bukowski\0\0\0\0\0\0\0\0\0\00
5: 9780312577220\0Fly Away (Firefly Lane, \0Kristin
                           Hannah\0\0\0\0\0\0\0\0\0\0\00
```

```
6: 9781620610090\0Opal (Lux, #3)\0\0\0\0\0\0\0\0\0\0\0Jennifer L.
Armentrout\0\0\00
```

Return value in $v0: -1

Resulting `books` contents:

```
7 7 68 (capacity, size, element_size)
0: 9780553214830\0The Declaration of Indep\0Founding
Fathers\0\0\0\0\0\0\0\0\00
1: 9780446695660\0Double Whammy\0\0\0\0\0\0\0\0\0\0\0Carl
                          Hiaasen\0\0\0\0\0\0\0\0\0\0\0\00
2: 9780060855900\0Equal Rites (Discworld, \0Terry
                          Pratchett\0\0\0\0\0\0\0\0\0\00
3: 9780345527780\0Wicked Business (Lizzy &\0Janet
                          Evanovich\0\0\0\0\0\0\0\0\0\00
4: 9788433914260\0Hollywood\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0Charles
                          Bukowski\0\0\0\0\0\0\0\0\00
5: 9780312577220\0Fly Away (Firefly Lane, \0Kristin
                          Hannah\0\0\0\0\0\0\0\0\0\0\00
6: 9781620610090\0Opal (Lux, #3)\0\0\0\0\0\0\0\0\0\0\0Jennifer L.
                          Armentrout\0\0\00
```

Part 8: Compute the Hash Code for an (ISBN, Customer ID #) Pair in a Given Hash Table

```
int hash_booksale(Hashtable* sales, string isbn, int customer_id)
```

The `hash_booksale` function takes a pointer to a valid `Hashtable` struct named `sales`, and an ISBN-13 value called `isbn` and customer ID # called `customer_id` for which we want to compute the hash function. The hash function itself is very simple: sum the ASCII codes of the 13 characters in `isbn` and the digits of `customer_id`, and then take the modulus of dividing the grand total by the capacity of the `sales` hash table (e.g., sum of all the ASCII codes from isbn and the `digits` from `customer_id` *mod* the capacity).

The function takes the following arguments, in this order:

- `books`: the starting address of a valid `Hashtable` of `Book` structs
- `isbn`: a 13-character, null-terminated string that represents a valid ISBN
- `customer_id`: an integer providing the customer ID#

Returns in $v0:

- The value of the hash function when evaluated for the provided ISBN and customer ID#.

Additional requirements:

- The function may not write any changes to main memory.

Example #1:

`hashtable` = pointer to a valid hash table with capacity = 11
`isbn = "9780671028370"`
`customer_id = 829273`
Return value in $v0: 9

Example #2:

`hashtable` = pointer to a valid hash table with capacity = 13
`isbn = "9780071401940"`
`customer_id = 93730282`
Return value in $v0: 6

Example #3:

`hashtable` = pointer to a valid hash table with capacity = 19
`isbn = "9712312301940"`
`customer_id = 512312`
Return value in $v0: 15

Part 9: Determine if a Year is a Leap Year

```
int is_leap_year(int year)
```

The `is_leap_year` function does what it sounds like - it determines if a given year is a leap year. A year is a leap year if it is after 1582 and it is divisible by 4, except centenary years not divisible by 400 (1700, 1800, 1900, 2100, etc.) Feel free to adapt the sample code given in the lecture slides and course materials that implements the logic needed for this function

The function takes one argument:
- `year`: an integer representing...a year

Returns in $v0:
- 0 if `year` is < 1582; or
- 1 if `year` is a leap year; or
- the number of years *negated* until the next leap year if `year` is not a leap year

Additional requirements:
- The function may not make any changes to main memory.

Function Argument	**Return Value**
`1244`	`0`

1900	-4
2000	1
5602	-2
2997	-7
3997	-3

Part 10: Convert a String Representation of a Time Interval to an Integer Number of Days

```
int datestring_to_num_days(string start_date, string end_date)
```

The `datestring_to_num_days` function takes two dates given as strings, in the format YYYY-MM-DD, and returns the number of days that elapse from `start_date` to `end_date`. For example, if `start_date = "2020-10-01"` and `end_date = "2020-10-10"`, the return value will be 9. Note that `start_date` itself is not included in the number of days elapsed. The function must account for leap years by calling the `is_leap_year` function. You may assume that both arguments are valid.

The function takes the following arguments, in this order:

- `start_date`: a date given in the format YYYY-MM-DD that is 1600-01-01 or later
- `end_date`: a date given in the format YYYY-MM-DD that is 1600-01-01 or later

Returns in $v0:

- The number of days elapsed between `start_date` and `end_date`, including `end_date`, but excluding `start_date`; or
- -1 if `end_date` is before `start_date`

Additional requirements:

- The function may not write any changes to main memory.
- ~~The function must call `is_leap_year`.~~ Not required because at least one student is using the Julian calendar to calculate this!

Function Arguments	**Return Value**
`"2020-10-12", "2020-10-25"`	13
`"2020-10-25", "2020-10-12"`	-1
`"2019-10-25", "2020-10-25"`	366
`"1692-11-28", "2602-09-13"`	332293
`"2019-01-08", "8410-09-08"`	2334508

`"3156-05-08", "3156-05-08"`	`0`

Part 11: Record the Sale a Book

```
int, int sell_book(Hashtable* sales, Hashtable* books, string isbn,
                   int customer_id, string sale_date, int sale_price)
```

The `sell_book` function records the sale of a book by adding an entry into the hash table `sales` of `BookSale` objects and updating the `times_sold` count of the sold book. We will make the following assumptions for this function:

- Deletions from the sales hash table never take place. Therefore, every entry in `sales.elements` is either all zeros (indicating an *empty* entry) or a valid `BookSale` struct.
- No customer ever buys the same book twice.
- The store has an infinite number of copies of every book.

Note that `sell_book` is similar in spirit to the `add_book` function from earlier in the assignment. The `sell_book` function attempts to insert a new `BookSale` struct into a given `sales` hash table by initializing the contents of the struct with the provided ISBN, customer ID# and sale price. The value stored in the struct for `sale_date` is provided by the return value of `datestring_to_num_days("1600-01-01", sale_date)`. `sell_book` begins by checking if the `sales` hash table is full. If so, the function returns (-1, -1) and makes no changes to the hash table's contents. It also checks if a book with the given ISBN exists in the `books` hash table. If not, the function returns (-2, -2). Otherwise, the function attempts to insert the struct at `sales.elements[hash_booksale(sales, isbn, customer_id)]`. Failing that, it performs linear probing to find the next *empty* entry in the `sales.elements` array and inserts the struct at that location. If needed, probing wraps-around from the last entry in `elements` to the first. The value of the `times_sold` field of the corresponding book in the books hash table must be incremented by 1. The function returns in $v0 the index where the struct was inserted. It returns in $v1 the number of entries in elements it had to read before finding an empty entry. For example, if it must skip over 3 occupied slots before finding an empty one, $v1 would be 4 in that case. If a sale is successfully added to the hash table, `sales.size` is incremented by 1.

The function takes the following arguments, in this order:

- `sales`: the starting address of a valid `Hashtable` of `BookSale` structs
- `books`: the starting address of a valid `Hashtable` of `Book` structs
- `isbn`: a 13-character, null-terminated string that represents a valid ISBN
- `customer_id`: an integer that represents a customer's ID#
- `sale_date`: a 10-character, null-terminated string that represents a valid date on or after January 1, 1600. Available at 0($sp). Assume the date string is value and represents a date on or after January 1, 1600.
- `sale_price`: an integer that represents the price paid for the book. Available at 4($sp).

Returns in $v0:

- The index in `elements` where the new `BookSale` struct was added to the `sales` hash table; or
- -1 if the hash table was full before the function call or
- -2 if no book of the given ISBN exists in the `books` hash table.

Returns in $v1:
- The number of entries in the `elements` array that were accessed before the `BookSale` struct was added to the `sales` hash table; or
- -1 if the hash table was full before the function call; or
- -2 if no book of the given ISBN exists in the `books` hash table.

Additional requirements:
- The function may only change the contents of the `books` and `sales` hash tables and no other parts of memory.
- The function must call `get_book`, `datestring_to_num_days`, `hash_booksale`, and `memcpy`.

Example #1: `sales` hash table is empty

```
isbn = "9780670032080"
customer_id = 12345
sale_date = "2020-09-14"
sale_price = 50
```
Contents of `sales` hash table:
```
9 0 28 (capacity, size, element_size)
0: empty
1: empty
2: empty
3: empty
4: empty
5: empty
6: empty
7: empty
8: empty
```
Contents of `books` hash table:
```
7 6 68 (capacity, size, element_size)
0: 9780345501330\0Fairy Tail, Vol. 1 (Fair\0Hiro Mashima, William Fl\00
1: empty
2: 9780060855900\0Equal Rites (Discworld, \0Terry
                                    Pratchett\0\0\0\0\0\0\0\0\0\00
3: 9780670032080\0Financial Peace Revisite\0Dave
                                    Ramsey\0\0\0\0\0\0\0\0\0\0\0\0\00
4: 9780064408330\0Joey Pigza Swallowed the\0Jack
                                    Gantos\0\0\0\0\0\0\0\0\0\0\0\0\00
5: 9780312577220\0Fly Away (Firefly Lane, \0Kristin
                                  Hannah\0\0\0\0\0\0\0\0\0\0\00
6: 9781416971700\0Out of My Mind\0\0\0\0\0\0\0\0\0\0\0Sharon M.
                                   Draper\0\0\0\0\0\0\0\0\00
```

Return value in $v0: 5
Return value in $v1: 1
Resulting `sales` contents:

```
9 1 28 (capacity, size, element_size)
0: empty
1: empty
2: empty
3: empty
4: empty
```
`5: 9780670032080\0 0 0 12345 153659 50`
```
6: empty
7: empty
8: empty
```

Resulting `books` contents:

```
7 6 68 (capacity, size, element_size)
0: 9780345501330\0Fairy Tail, Vol. 1 (Fair\0Hiro Mashima, William Fl\00
1: empty
2: 9780060855900\0Equal Rites (Discworld, \0Terry
                                  Pratchett\0\0\0\0\0\0\0\0\0\00
3: 9780670032080\0Financial Peace Revisite\0Dave
                                  Ramsey\0\0\0\0\0\0\0\0\0\0\0\0\0\01
4: 9780064408330\0Joey Pigza Swallowed the\0Jack
                                  Gantos\0\0\0\0\0\0\0\0\0\0\0\0\0\00
5: 9780312577220\0Fly Away (Firefly Lane, \0Kristin
                                  Hannah\0\0\0\0\0\0\0\0\0\0\00
6: 9781416971700\0Out of My Mind\0\0\0\0\0\0\0\0\0\0\0Sharon M.

Draper\0\0\0\0\0\0\0\0\00
```

Example #2: `sales` hash table contains some entries; a few steps of linear probing required to insert the `BookSale` struct

```
isbn = "9780670032080"
customer_id = 6123
sale_date = "2019-11-04"
sale_price = 1032
```

Contents of `sales` hash table:

```
9 4 28 (capacity, size, element_size)
0: empty
1: 9781416971700\0 0 0 2323432 155136 22
2: 9780060855900\0 0 0 920192 158715 61
3: 9780345501330\0 0 0 81321 151369 192
```

```
4: empty
5: 9780312577220\0 0 0 2424 152013 125
6: empty
7: empty
8: empty
```
Contents of `books` hash table:
```
7 6 68 (capacity, size, element_size)
0: 9780345501330\0Fairy Tail, Vol. 1 (Fair\0Hiro Mashima, William Fl\01
1: empty
2: 9780060855900\0Equal Rites (Discworld, \0Terry
                                     Pratchett\0\0\0\0\0\0\0\0\0103
3: 9780670032080\0Financial Peace Revisite\0Dave
                                     Ramsey\0\0\0\0\0\0\0\0\0\0\0\0\061
4: 9780064408330\0Joey Pigza Swallowed the\0Jack
                                     Gantos\0\0\0\0\0\0\0\0\0\0\0\0\044
5: 9780312577220\0Fly Away (Firefly Lane, \0Kristin
                                     Hannah\0\0\0\0\0\0\0\0\0\0\0812
6: 9781416971700\0Out of My Mind\0\0\0\0\0\0\0\0\0\0\0Sharon M.
                                     Draper\0\0\0\0\0\0\0\0\01
```

Return value in $v0: 4
Return value in $v1: 3
Resulting `sales` contents:
```
9 5 28 (capacity, size, element_size)
0: empty
1: 9781416971700\0 0 0 2323432 155033 22
2: 9780060855900\0 0 0 920192 158610 61
3: 9780345501330\0 0 0 81321 151269 192
4: 9780670032080\0 0 0 6123 153344 1032
5: 9780312577220\0 0 0 2424 151912 125
6: empty
7: empty
8: empty
```
Resulting `books` contents:
```
7 6 68 (capacity, size, element_size)
0: 9780345501330\0Fairy Tail, Vol. 1 (Fair\0Hiro Mashima, William Fl\01
1: empty
2: 9780060855900\0Equal Rites (Discworld, \0Terry
                                     Pratchett\0\0\0\0\0\0\0\0\0103
3: 9780670032080\0Financial Peace Revisite\0Dave
                                     Ramsey\0\0\0\0\0\0\0\0\0\0\0\0\062
4: 9780064408330\0Joey Pigza Swallowed the\0Jack
                                     Gantos\0\0\0\0\0\0\0\0\0\0\0\0\044
5: 9780312577220\0Fly Away (Firefly Lane, \0Kristin
                                     Hannah\0\0\0\0\0\0\0\0\0\0\0812
```

```
6: 9781416971700\0Out of My Mind\0\0\0\0\0\0\0\0\0\0\0Sharon M.
                                    Draper\0\0\0\0\0\0\0\0\01
```

Example #3: provided ISBN does not exist in the `books` hash table

```
isbn = "9780679744400"
customer_id = 5123
sale_date = "2029-11-04"
sale_price = 987
```

Contents of `sales` hash table:

```
9 4 28 (capacity, size, element_size)
0: empty
1: 9781416971700\0 0 0 2323432 155136 22
2: 9780060855900\0 0 0 920192 158715 61
3: 9780345501330\0 0 0 81321 151369 192
4: empty
5: 9780312577220\0 0 0 2424 152013 125
6: empty
7: empty
8: empty
```

Contents of `books` hash table:

```
7 6 68 (capacity, size, element_size)
0: 9780345501330\0Fairy Tail, Vol. 1 (Fair\0Hiro Mashima, William Fl\01
1: empty
2: 9780060855900\0Equal Rites (Discworld, \0Terry
                                    Pratchett\0\0\0\0\0\0\0\0\0\0103
3: 9780670032080\0Financial Peace Revisite\0Dave
                                    Ramsey\0\0\0\0\0\0\0\0\0\0\0\0\0\061
4: 9780064408330\0Joey Pigza Swallowed the\0Jack
                                    Gantos\0\0\0\0\0\0\0\0\0\0\0\0\0\044
5: 9780312577220\0Fly Away (Firefly Lane, \0Kristin
                                    Hannah\0\0\0\0\0\0\0\0\0\0\0812
6: 9781416971700\0Out of My Mind\0\0\0\0\0\0\0\0\0\0\0Sharon M.
                                    Draper\0\0\0\0\0\0\0\0\01
```

Return value in $v0: -2
Return value in $v1: -2
Resulting `sales` contents:

```
Resulting sales contents:
9 4 28 (capacity, size, element_size)
0: empty
1: 9781416971700\0 0 0 2323432 155033 22
2: 9780060855900\0 0 0 920192 158610 61
3: 9780345501330\0 0 0 81321 151269 192
4: empty
```

```
5: 9780312577220\0 0 0 2424 151912 125
6: empty
7: empty
8: empty
```
Resulting `books` contents:
```
7 6 68 (capacity, size, element_size)
0: 9780345501330\0Fairy Tail, Vol. 1 (Fair\0Hiro Mashima, William Fl\01
1: empty
2: 9780060855900\0Equal Rites (Discworld, \0Terry
                                     Pratchett\0\0\0\0\0\0\0\0\0\0103
3: 9780670032080\0Financial Peace Revisite\0Dave
                                     Ramsey\0\0\0\0\0\0\0\0\0\0\0\0\0\061
4: 9780064408330\0Joey Pigza Swallowed the\0Jack
                                     Gantos\0\0\0\0\0\0\0\0\0\0\0\0\0\044
5: 9780312577220\0Fly Away (Firefly Lane, \0Kristin
                                     Hannah\0\0\0\0\0\0\0\0\0\0\0812
6: 9781416971700\0Out of My Mind\0\0\0\0\0\0\0\0\0\0\0Sharon M.
                                     Draper\0\0\0\0\0\0\0\0\0\01
```

Example #4: the `sales` hash table is full

```
isbn = "9781416971700"
customer_id = 1523
sale_date = "2022-10-14"
sale_price = 323
```
Contents of `sales` hash table:
```
9 9 28 (capacity, size, element_size)
0: 9780345501330\0 0 0 723341 155332 55
1: 9781416971700\0 0 0 2323432 155136 22
2: 9780060855900\0 0 0 920192 158715 61
3: 9780345501330\0 0 0 81321 151369 192
4: 9780312577220\0 0 0 777233 155332 55
5: 9780312577220\0 0 0 2424 152013 125
6: 9780345501330\0 0 0 26234 155332 55
7: 9780312577220\0 0 0 12312 155332 55
8: 9780064408330\0 0 0 73123 155332 55
```
Contents of `books` hash table:
```
7 6 68 (capacity, size, element_size)
0: 9780345501330\0Fairy Tail, Vol. 1 (Fair\0Hiro Mashima, William Fl\03
1: empty
2: 9780060855900\0Equal Rites (Discworld, \0Terry
                                     Pratchett\0\0\0\0\0\0\0\0\0\0103
3: 9780670032080\0Financial Peace Revisite\0Dave
                                     Ramsey\0\0\0\0\0\0\0\0\0\0\0\0\0\061
4: 9780064408330\0Joey Pigza Swallowed the\0Jack
```

```
                                     Gantos\0\0\0\0\0\0\0\0\0\0\0\0\0\045
5: 9780312577220\0Fly Away (Firefly Lane, \0Kristin
                                     Hannah\0\0\0\0\0\0\0\0\0\0\0814
6: 9781416971700\0Out of My Mind\0\0\0\0\0\0\0\0\0\0\0Sharon M.
                                     Draper\0\0\0\0\0\0\0\0\0\01
```

Return value in $v0: -1
Return value in $v1: -1
Resulting `sales` contents:

```
9 9 28 (capacity, size, element_size)
0: 9780345501330\0 0 0 723341 155229 55
1: 9781416971700\0 0 0 2323432 155033 22
2: 9780060855900\0 0 0 920192 158610 61
3: 9780345501330\0 0 0 81321 151269 192
4: 9780312577220\0 0 0 777233 155229 55
5: 9780312577220\0 0 0 2424 151912 125
6: 9780345501330\0 0 0 26234 155229 55
7: 9780312577220\0 0 0 12312 155229 55
8: 9780064408330\0 0 0 73123 155229 55
```

Resulting `books` contents:

```
7 6 68 (capacity, size, element_size)
0: 9780345501330\0Fairy Tail, Vol. 1 (Fair\0Hiro Mashima, William Fl\03
1: empty
2: 9780060855900\0Equal Rites (Discworld, \0Terry
                                Pratchett\0\0\0\0\0\0\0\0\0\0103
3: 9780670032080\0Financial Peace Revisite\0Dave
                                Ramsey\0\0\0\0\0\0\0\0\0\0\0\0\0\061
4: 9780064408330\0Joey Pigza Swallowed the\0Jack
                                Gantos\0\0\0\0\0\0\0\0\0\0\0\0\0\045
5: 9780312577220\0Fly Away (Firefly Lane, \0Kristin
                                Hannah\0\0\0\0\0\0\0\0\0\0\0814
6: 9781416971700\0Out of My Mind\0\0\0\0\0\0\0\0\0\0\0Sharon M.
                                Draper\0\0\0\0\0\0\0\0\0\01
```

In the remainder of the assignment we will explore an interesting computational problem described on geeksforgeeks.com. In our case, we will optimize the sale of books, rather than wines, which is used in the geeksforgeeks problem

Part 12: Compute the Total Sales for a Given Ordering of Books

```
int compute_scenario_revenue(BookSale[] sales_list, int num_sales,
                             int scenario)
```

The `compute_scenario_revenue` function takes an array of at most 32 `BookSale` structs and, according to the bitstring `scenario`, computes the total revenue generated by selling the given books in that order, with the small twist that as time goes on, the revenue realized from the sale of a book actually increases. The integer `scenario` encodes the order in which the books are sold. **Note:** for the purposes of this function we will ignore the `sale_date` field in each `BookSale` struct. For instance, suppose 5 `BookSale` structs are stored in the `sales_list` array, and `int scenario = 12`. The `scenario` bitstring 01100, read left-to-right, would mean: sell the leftmost book (and remove it), then sell the rightmost book (and remove it), then sell the rightmost book (and remove it), then sell the leftmost book (and remove), and then sell the leftmost book.

As an example, suppose our list of BookSale structs had the following corresponding ISBNs:

```
sales_list[0]: 0000000000000
sales_list[1]: 1111111111111
sales_list[2]: 2222222222222
sales_list[3]: 3333333333333
sales_list[4]: 4444444444444
```

The bitstring 01100 indicates that the books would be sold in the following order:

```
1. 0000000000000   (leftmost book)
2. 4444444444444   (rightmost book remaining)
3. 3333333333333   (rightmost book remaining)
4. 1111111111111   (leftmost book remaining)
5. 2222222222222   (final book)
```

Note that we are reading the bitstring from left-to-right, not right-to-left!

Now, the significance of this order becomes apparent when we see what happens when we sell a book. Imagine that each book is sold on a different day (again, remember that we are ignoring the `sale_date` field of the `BookSale` structs for this problem). On Day 1, the chosen book is sold at its normal price, as given by its `sale_price`. On Day 2, the next book is sold at 2x its `sale_price`. On Day 3, the next book is sold at 3x its `sale_price`, and so on. The n-th book is sold at n times its `sale_price`.

Let's go back to our example. Suppose the books have the indicated prices: stored in the `sale_price` field of each struct.

```
sales_list[0]: 0000000000000  $2
sales_list[1]: 1111111111111  $4
sales_list[2]: 2222222222222  $6
sales_list[3]: 3333333333333  $2
sales_list[4]: 4444444444444  $5
```

We have our bitstring 01100 that indicates the selling order. As time goes on, we can see how the total revenue of selling the books in that order is realized:

1. Sell `0000000000000        $2   Total: 1*$2 = $2`
2. Sell `4444444444444  $5   Total: $2 + 2*$5 = $12`
3. Sell `3333333333333  $2   Total: $12 + 3*$2 = $18`
4. Sell `1111111111111  $4   Total: $18 + 4*$4 = $34`
5. Sell `2222222222222  $6   Total: $34 + 5*$6 = $64`

For this particular selling scenario, the total revenue is $64.

The function takes the following arguments, in this order:

- `sales_list`: a completely filled array of `BookSale` structs
- `num_sales`: the number of BookSale structs in `sales_list`
- `scenario`: a 32-bit integer that encodes the order in which to sell the books in `sales_list`

Returns in $v0:

- The total revenue produced by selling the books in the order specified by `scenario`.

Additional requirements:

- The function may not write any changes to main memory.

Example #1:

```
num_sales = 5
scenario = 00000000000000000000000000001100
```

Contents of `sales_list`:

```
0000000000000\0 0 0 12345 153789 2
1111111111111\0 0 0 52321 153789 4
2222222222222\0 0 0 41231 153789 6
3333333333333\0 0 0 12523 153789 2
4444444444444\0 0 0 51231 153789 5
```

Return value in $v0: 64

Example #2:

```
num_sales = 9
scenario = 00000000000000000000000011001101
```

Contents of `sales_list`:

```
5688198170802\0 0 0 69170 1836 54
8174611363470\0 0 0 42817 1416 472
6771105776755\0 0 0 73451 1537 296
9694257705423\0 0 0 89692 1796 348
8148291072965\0 0 0 93505 1860 115
1537177706509\0 0 0 25366 1488 433
```

```
3918738489597\0 0 0 52178 1247 303
1856189180494\0 0 0 69428 1731 118
4206531586624\0 0 0 99857 1083 190
```
Return value in $v0: 12824

Example #3:

```
num_sales = 11
scenario = 000000000000000000000**01101001101**
```
Contents of `sales_list`:
```
0043444591466\0 0 0 56274 1935 12
0912549927792\0 0 0 20150 1615 341
8208666153502\0 0 0 13570 1727 275
7891835066590\0 0 0 89612 1679 405
7850924139905\0 0 0 56805 1379 227
7172692040044\0 0 0 14473 1609 306
8881948157382\0 0 0 35983 1662 38
6319285981025\0 0 0 61086 1664 411
5218182478429\0 0 0 34089 1400 484
9545006833495\0 0 0 12400 1382 458
6178171431772\0 0 0 40266 1773 330
```
Return value in $v0: 19581

Example #4:

```
num_sales = 13
scenario = 00000000000000000001101101001101
```
Contents of `sales_list`:
```
3318220944005\0 0 0 68169 1310 429
1790274371133\0 0 0 12055 1911 67
2323508691228\0 0 0 25027 1950 382
6580406620516\0 0 0 14457 1160 390
1220293838689\0 0 0 26212 1181 79
8838065641525\0 0 0 38406 1406 239
4140073652158\0 0 0 91560 1707 231
0358003692171\0 0 0 97221 1131 492
8102823570747\0 0 0 64311 1671 147
2694841805861\0 0 0 81769 1342 380
8985832671846\0 0 0 33475 1157 455
9649316844603\0 0 0 74921 1199 446
2768474273172\0 0 0 48218 1291 474
```
Return value in $v0: 25886

Part 13: Find the Optimal Selling Order to Maximize Revenue

```
int maximize_revenue(BookSale[] sales_list, int num_sales)
```

The `maximize_revenue` function takes an array of `BookSale` objects and determines in which order to sell the books to maximize the revenue. As laid out in the previous part of the assignment, we imagine a situation in which each day we select one book from the list of books and sell it. However, the next book to sell may be taken only from the front or the rear of the array.

Although the geeksforgeeks web page recommends using recursion to solve this problem, you are not expected or required to do so. In fact, a simple, iterative brute-force algorithm can be applied to solve this problem. Suppose `n` is the number of `BookSale` structs in `sales_list`. We can simply enumerate all 2^n possible bitstrings from 0 to 2^n-1 and compute the revenues realized by selling the books in every possible order. In other words, we call `compute_scenario_revenue(sales_list, num_sales, bitstring)` for each of the 2^n possible bitstrings and choose the maximum returned value. That's it!

Enumerating the 2^n possible bitstrings is very simple: in a for-loop, just count from 0 to 2^n-1 and pass each of those values as `scenario` to the `compute_scenario_revenue` function. For large test cases, the maximum MIPS instruction count will be appropriately increased.

The function takes the following arguments, in this order:

- `sales_list`: a completely filled array of `BookSale` structs
- `num_sales`: the number of BookSale structs in `sales_list`

Returns in $v0:

- The maximum revenue that can be realized by selling the books in `sales_list` as described in this part and the previous part of the assignment.

Additional requirements:

- The function may not write any changes to main memory.
- The function must call `compute_scenario_revenue`.

Example #1:

`num_sales = 5`
Contents of `sales_list`:

```
0000000000000\0 0 0 12345 153789 2
1111111111111\0 0 0 52321 153789 4
2222222222222\0 0 0 41231 153789 6
3333333333333\0 0 0 12523 153789 2
4444444444444\0 0 0 51231 153789 5
```

Return value in $v0: 64

Example #2:

```
num_sales = 9
```
Contents of `sales_list`:
```
0845558347906\0 0 0 65818 1620 82
5577045702462\0 0 0 91689 1951 154
6354780489355\0 0 0 94530 1579 60
1999320995468\0 0 0 93964 1715 225
0145174318443\0 0 0 41570 1195 232
5871544817889\0 0 0 57193 1139 387
7106045480035\0 0 0 48631 1282 414
1730871923235\0 0 0 39311 1399 78
8122589552824\0 0 0 60637 1888 497
```
Return value in $v0: 12980

Example #3:

```
num_sales = 11
```
Contents of `sales_list`:
```
5289996563210\0 0 0 50402 1912 114
4878656880115\0 0 0 14181 1025 40
1545953167947\0 0 0 74773 1453 419
4163887205026\0 0 0 11881 1088 316
3994954632401\0 0 0 25708 1418 346
2840757350273\0 0 0 79850 1501 89
2690573805693\0 0 0 75967 1093 147
6412500793328\0 0 0 20379 1997 380
1956670228687\0 0 0 95494 1343 275
4719224196873\0 0 0 30863 1956 194
1049903854879\0 0 0 40347 1980 281
```
Return value in $v0: 18092

Example #4:

```
num_sales = 13
```
Contents of `sales_list`:
```
0969867337768\0 0 0 91339 1799 274
7167926219358\0 0 0 87431 1648 149
5856271181125\0 0 0 38069 1171 135
1739956344089\0 0 0 24120 1716 401
2423634816035\0 0 0 44091 1735 429
7323738407143\0 0 0 33518 1687 357
0790885414601\0 0 0 49219 1389 468
2095877382616\0 0 0 89257 1679 325
7788346465697\0 0 0 76118 1374 364
7903951955477\0 0 0 45362 1032 44
2668430009430\0 0 0 26427 1969 245
```

```
5802412331328\0 0 0 94001 1133 62
0689148237261\0 0 0 28651 1412 265
```
Return value in $v0: 29443

Academic Honesty Policy

Academic honesty is taken very seriously in this course. By submitting your work for grading you indicate your understanding of, and agreement with, the following Academic Honesty Statement:

1. I understand that representing another person's work as my own is academically dishonest.
2. I understand that copying, even with modifications, a solution from another source (such as the web or another person) as a part of my answer constitutes plagiarism.
3. I understand that sharing parts of my homework solutions (text write-up, schematics, code, electronic or hard-copy) is academic dishonesty and helps others plagiarize my work.
4. I understand that protecting my work from possible plagiarism is my responsibility. I understand the importance of saving my work such that it is visible only to me.
5. I understand that passing information that is relevant to a homework/exam to others in the course (either lecture or even in the future!) for their private use constitutes academic dishonesty. I will only discuss material that I am willing to openly post on the discussion board.
6. I understand that academic dishonesty is treated very seriously in this course. I understand that the instructor will report any incident of academic dishonesty to the University's Academic Judiciary.
7. I understand that the penalty for academic dishonesty might not be immediately administered. For instance, cheating on a homework assignment may be discovered and penalized after the grades for that homework have been recorded.
8. I understand that buying or paying another entity for any code, partial or in its entirety, and submitting it as my own work is considered academic dishonesty.
9. I understand that there are no extenuating circumstances for academic dishonesty.

How to Submit Your Work for Grading

To submit your hwk4.asm file for grading:

1. Go to the course website.
2. Click the Submit link for this assignment.
3. Type your SBU ID# on the line provided.
4. Press the button marked **Add file** and follow the directions to attach your file.
5. Hit Submit to submit your file grading.

Oops, I messed up and I need to resubmit a file!

No worries! Just follow the steps again. We will grade only your last submission.