CSC 236 T9: Binary Addition

**This is a team assignment designed as an in-class activity related to binary and linked lists.**

Note that this team assignment was modified from an assignment created by <http://cis1.towson.edu/~cssecinj/modules/cs0/>.

**Directions for use:**

* To use this form effectively, sign into a Google account.
* Then under “File” choose “Make a Copy” in order to be able to edit.
* Share with all team members, but allow Recorder to do the recording.
* Each yellow box should be filled with an appropriate team response..
* Download as *yourteamname-t9.docx* and upload to Moodle

First, rotate and confirm the new roles and complete the form below for assigned roles of each member.  Try to assign a role to each member that they have not yet had.

**Member Roles**

* If you have only four people, combine Quality Control Officer & Process Analyst
* If you have only three people, also combine Recorder & Spokesperson

|  |  |
| --- | --- |
| **Team Roles** | **Member Name** |
| **Facilitator:** | **Cody Grinnell** |
| **Recorder:** | **John Hellrung** |
| **Spokesperson:** | **Zach Ball** |
| **Quality Control Officer:** | **Angie Li** |
| **Process Analyst:** | **Angie Li** |

***Binary Numbers in Linked Lists***

Recall that we have been working with a BinaryNumber class and a related Bit class which stores a binary number "backwards" in the linked list with the more significant bits are further down the list, so 1011 is stored as ➔1➔1➔0➔1➔None.

1011 in binary is 1x23 +0x22 +1x21 +1x20 =8+0+2+1 which is eleven in decimal.

If we wish to add 1 to this number, we should get twelve in decimal. Since 1+ 1 =10 in binary, adding 1 to 1011 causes the carry to ripple to the left until we get 1100 in binary.

1100 is 1x23 +1x22 +0x21 +0x20 =8+4+0+0  which is twelve.

|  |  |  |  |
| --- | --- | --- | --- |
| **binary of N** | **binary of N+1** | **decimal of N** | **decimal of N+1** |
| 1011 | 1100 | 11 (eleven) | 12 (twelve) |
| 100 | 101 | 4 | 5 |
| 110 | 111 | 6 | 7 |
| 111 | 1000 | 7 | 8 |

***Boundary Conditions or Edge Cases***

The terms "boundary conditions" or "edge cases" are often used when dealing with the designing and testing of algorithms and their implementations. When designing or testing an algorithm, it is important to pay attention to the "boundaries" or "edges" of the input. In programming, an edge case often involves input values that require special handling.  They are often the values you want to use in testing so you can make sure to have good coverage in your testing suite,

Identify the main boundary values or edge cases for **incrementing** (a.k.a adding 1 to) a binary number N stored backwards as a linked list, and identify any special handling which will be required in the implementation.

|  |  |
| --- | --- |
| **Boundary Values / Edge Cases for N+1 when N is a binary number** | **Special handling?** |
| Boundary existed when binary number consist of only ones. | All of the 1 values become 0s and an additional 1 is appended to the n+1 index (the last). |

***Adding Two Binary Numbers (A + B)***

Adding two binary numbers is in some senses a generalization of incrementing, and in some senses is differs.

Consider adding the binary numbers 111 + 10 which is 7 + 2 = 9 in decimal.

 111

+ 10

----

1001

Add the following binary numbers:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **binary A** | **binary B** | **binary A+B** | **decimal A** | **decimal B** | **decimal A+B** |
| 111 | 10 | 1001 | 7 | 2 | 9 |
| 110 | 1 | 111 | 6 | 1 | 7 |
| 110 | 101 | 1011 | 6 | 5 | 11 |
| 111 | 1111 | 10110 | 7 | 15 | 22 |

***Boundary conditions for adding two Binary Numbers (A + B)***

Identify the main boundary values or edge cases for **adding two binary numbers A+B** when the numbers are stored backwards as linked lists, and any special handling these cases necessitate.

|  |  |
| --- | --- |
| **Boundary Values / Edge Cases for A + B** | **special handling?** |
| Edge cases exist when you have to carry a 1 into a sum of two 1s.  For example:  1011 + 111 | Using the modulus operator % and integer division, we can determine what needs the sum should be and what needs to be carried.  Ex. 1011 + 111 -> When adding the second 1 and 1 with a 1 in the carry, we have 3%2=1 and 3/2 = 1. So we put 1 down and carry 1 since two fully went into 3 and 1 was left over. |

***Algorithm design for C=A+B when A, B, and C are binary numbers***

Design a pseudocode algorithm which could be implemented in Python, C++ or another language given that A, B, and C are all linked lists which store bits “backwards.”

i.e. 1011 is stored as ➔1➔1➔0➔1➔None.

Make sure that this design is not specific to Python or to C++ and should use only conceptual ideas.

Indicate functions that encapsulate specific operations with details how they will work, such as pre and post conditions.  It is not acceptable to do the entire solution in a single function.

|  |
| --- |
| **Pseudocode design for binary C=A+B** |
| Create a function that makes both A & B have the same length.    pre: A and B are linked lists containing bits     post: A and B are the same size linked lists  Function for adding two bits (D and E) together that keep track of the sum and the carry value.     pre: D and E are bits from a binaryNumber object.     post: returns a tuple containing the new place value as well as if there is a carry ( Ex. (True,    False) means the new place value is 1 and there is no carry) |
|  |

**Suggestions and Submission**

Please offer any suggestions for improvement of this activity from the team:

|  |
| --- |
| **Suggestions for improvement** |
| Format the following table so that each of the columns alternate colors. It would be easier to interpret.   |  |  |  |  | | --- | --- | --- | --- | | **binary of N** | **binary of N+1** | **decimal of N** | **decimal of N+1** | | 1011 | 1100 | 11 (eleven) | 12 (twelve) | | 100 | 101 | 4 | 5 | | 110 | 111 | 6 | 7 | | 111 | 1000 | 7 | 8 | |

To submit, the Recorder will download as *yourteamname-t9.docx* and upload to Moodle while all other members will simply upload the name of the assignment (t9) and the names and roles of all team members.