Meet Hitesh Sonagara

122208226

msonagara@mySeneca.ca  
Click or tap here to enter text.

*There are many notes in the instructions to help you earn marks for the questions below.*

Exercise One of Two – **integer overflow** (80 points)

1) 🡺 (7.5 points) ) If a variable counting seconds is stored in a signed **long** 32-bit integer, how many **days** will it take until that integer overflows? (to one decimal place)

It will take 248.54 days until it overflows.

2) 🡺 (15 points) Convert the maximum value of an unsigned **long** 32-bit integer, representing hundredths of a second, into whole numbers of  
 days : hours : minutes : seconds . hundredths of a second.   
After *n* days, how many hours remain? After *n* hours, how many minutes remain? etc.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **248** | **5,966** | **357,914** | **21,474,837** | **2,147,483,647** |
| **DAYS** | **HOURS :** | **MINUTES :** | **SECONDS .** | **HUNDREDTHS** |

3) 🡺 (2.5 points) What are the maximum and minimum values that can be stored in a **short** 16-bit signed integer?

16-bit signed integer maximum = 32,767 … minimum = -32,768

4) 🡺 (5+5 points) Give examples of two **short** 16-bit signed integers that when added together would cause overflow.

 17256 + 15999 are two positive values causing overflow when added together.

(-18347) + (-16784) are two negative values causing overflow when added together.

Binary Search Bug

5) 🡺 (10 points) What is potentially wrong with the **(low + high) / 2** calculation to find the middle point? Under what conditions would the calculation go wrong?

This method can cause overflow when two huge integers are added. When (low + high) becomes greater than capacity memory, it causes the program to overflow.

In order to avoid such situation, we can change arithmatic operations to prevent the risk of losing data.

6) 🡺 (10 points) REWRITE themidcalculation to prevent overflow*from*mid = (low + high) / 2;*to*  **mid = low + (high-low) / 2 ;**

7) 🡺 (25 points)Write a 250+ word “reflection”(similar to a workshop in your programming class) describing the steps you used to develop and test your solution to the calculation bug.

The addition of high and low can sometimes cause an unexpected overflow as it exceeds the memory range even though both the integers are in memory range. This situation troubles us with unexpected output. The overflow occurs when the (low + high) gives the data that is too large to hold in a temporary memory. So in order to avoid such overflow, we need to change the arithmatic operations for finding the middle number.

Instead of using [ (low+high) / 2 ] we can use [ low + (high-low) / 2 ]. This is a good way to prevent overflow as subtraction of two integers cannot produce a very high value. It can execute effectively as there are no threats in it. The reason why this will not cause overflow is that the two values (low) and (high - low) will be in range of memory capacity so they can be stored in temporary memory.

Exercise Two of Two – **Numbering Systems and Conversions (20 points)**

8) 🡺 (10 points ) What is the hex value for these colours?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Red decimal | Green decimal | Blue decimal | Hex triplet | Colour Description |
| 15 | 245 | 231 | #0ff5e7 | Bright Aqua |
| 192 | 255 | 238 | #c0ffee | Arabica Mint |
| 208 | 13 | 30 | #d00d1e | Scarlet Splendour |
| 186 | 187 | 30 | #babb1e | Fallout Green |
| 126 | 164 | 112 | #7ea470 | Frog Hollow |

9) 🡺 (10 points)Fill in this chart as per the column headings

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hex triplet | Red decimal | Green decimal | Blue decimal | Describe the Final Colour *and* change the cell's background colour, i.e. R-click and see MS Word 'Shading' |
| #302432 | 48 | 36 | 50 | Obsidian Red |
| #204C02 | 35 | 76 | 2 | Lincoln Green |
| #D64A53 | 214 | 74 | 83 | Faded Red |
| #404891 | 64 | 72 | 145 | Galaxy Express |