Student Name SONI DEV ALPESHBHAI
Student Number 130759210
UserID dasoni4 @mySeneca.ca
Special Notes to Instructor 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 hundredths of a second is stored in a signed **`long`** 32-bit integer,
how many **days**, to two decimals, will it take until that integer overflows?

2,147,483,647 / 8,640,000 = 248.55 = 248 days will be taken to overflow for a 32 bit integer.

2) ➔ (15 points) Convert the maximum value of a signed **`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	5966	357914	21474837	2147483647

DAYS HOURS : MINUTES : SECONDS .HUNDREDTHS

After

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.

16384 + 16385 are two positive values causing overflow when added together.

(-17569) + (-17000) 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?

When we add these two numbers, it can possibly cause an arithmetic overflow because (low + high) is higher than the capacity of the memory. There is a way, through which we can actually change the arithmetic operations such that is safer and with lesser chance of losing data. For an instance, let's say we have the low value of 1,000,000,000 and the high value of 2,000,000,000. The sum of low and high (3,000,000,000) would overflow on a long 32-bit integer. The result of the calculation would be (- 852,516,353 / 2) = -426,258,176.5

6) ➔ (10 points) REWRITE the `mid` calculation to prevent overflow

from `mid = (low + high) / 2;`

to **`mid = low + (high – low) / 2;`**

`Also, mid = (low/2) + (high/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 high and low integers are in a range of specific values. Their addition sometimes exceeds the range and overflow occurs. This gives us much trouble with unexpected output. During the compilation, this could often give unpredictable threats to the output of the written program. If (low+high) can produce a value too large to hold in temporary memory then an overflow occurs.

In order to prevent that, we have to find an alternative way of it. So with lots of trial and error, I actually found a way out, which is the combination of the term (high-low) is well suited for this. As subtraction of two integers can never give a higher value. Although it is not an efficient way, but at least it is safe to use. As no threats are found in this, it is actually successful. The difference between these two will be stored in the temporary memory without any overflow. Data stored using this equation will also be kept safe from losing the content.

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
186	187	30	BABB1E	LIGHT OLIVE
192	255	238	C0FFEE	CYAN
208	13	30	D00D1E	RED
126	164	112	7EA470	OLIVE
15	245	231	0FF5E7	CYAN

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'
#404892	64	72	146	NAVY
#D64A52	214	74	82	LIGHT MAROON
#204C02	32	76	2	DARK GREEN
#302434	48	36	52	BLACK