🡺 Your name: **Jason Jun** Student No.: **126683200** UserID: **­­­­jjun10@mySeneca.ca**

**Activity 1** of 3 – **integer overflow** (50 points)

a) 🡺 If the timer value is stored in a signed **long** 32-bit integer,   
how many **days**, to two decimals, will it take until that integer overflows? **(5 points)**

It will take 248.55 days until the integer overflows.

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

16-bit signed integer maximum = 32767 … minimum = -32768

c) 🡺 Give examples of values that would cause overflow in positive and negative directions when two **short** 16-bit signed integers are added together. **(5 points)**

32000 + 768 are two positive **short** values causing overflow when added together.

-32000 + (-768) are two negative **short** values causing overflow when added together.

**Your task is to identify and fix the bug in that calculation of mid.**

d) 🡺 What is potentially wrong with the **(low + high) / 2** calculation to find the middle point? Under what conditions would the calculation go wrong? **(7.5 points)**For a demonstration of the problem, run **MidBugTest.exe** found in this week's zip file.

By having the calculation, (low + high), it could lead to cause an overflow. If high value was at max, then adding the low value would result in going beyond the maximum limit and the user would get different results. Moreover, if low value was at the minimum limit and the high value was a negative value, then when low and high are added together, it would go beyond the minimum limit and the user would get different results. For example, the maximum value that can be stored in a short 16-bit signed integer is 32767. If high value is 32000 and low value is 800 and the two values are added together, it would cause an overflow because 32800 is bigger than 32767 and would output an unwanted result.

e) 🡺 REWRITE the code to prevent overflow **(10 points)***from* **mid = (low + high) / 2;***to* mid = low + ((high – low) / 2);

f) 🡺 Describe the steps you used to develop and test your solution to the binary search bug. **For the full 20 points**, what were the details of your process from problem analysis to solution implementation? (This is like the reflection component in your C course workshop.)

A C program to test your new formula, **MidBugTest.c** is found in this week's zip file. The code demonstrates the bug. Change the **mid =** line of code to your new formula, compile, and run.

Before changing the formula of mid in the code, I decided to run the code and see what the problems were. Then using the output I got after running the code, I started to examine which areas had problems and thought about it a lot. However, there were too many numbers and words and started to become confusing, so I decided to use the debugging system within the Microsoft Visual Studio. From debugging, I found a major problem which was an overflow whenever the summed up result between high and low was over the maximum and minimum values. Then first I tried to fix the formula by changing it, where mid = ((high - low) / 2). However, it would still overflow the minimum values if high value was negative, after multiple runs of testing and debugging. Through going through changing formula many times and testings, I finally found what I needed to get and that is mid = low + ((high - low) / 2). With this I was able to get same result of finding the mid value as when I first used the formula ((high + low) / 2). What the new formula do is first find the middle value between high and low values by subtracting high to low values and divide by 2. Then adding the low value to the division result would give the middle value that includes the minimum value. When running the code with the formula, it returned with no error and was able to run cleanly with no overflow.

**Activity 2** of 3 – **Boolean logic (25 points)**

**🡺 For any given date, what is the Boolean logic to decide if you have to attend school during the current term?** (a semester is called a “term” at Seneca)

// Check #1 - Is today within the term margins (between startClasses and endClasses)?

if (today < startClasses || today > endClasses) {

printf("don't go\n");

return 0;

}

// Check #2 - Is today within the study week margins

if (today >= startStudyWeek || today <= endStudyWeek) {

printf("don't go\n");

return 0;

}

// Check #3 - Is today a holiday?

if (today == familyDay || today == goodFriday || today == victoriaDay ||

today == canadaDay || today == civicDay || today == thanksgivingDay) {

printf("don't go\n");

return 0;

}

// Check #4 - Do you have a class today?

if (classesByDay[todayDayOfWeek] != 0) {

printf("go\n");

return 0;

} else {

printf("don't go\n");

return 0;

}

**Activity 3a and 3b** of 3 – **Numbering Systems and Conversions (15 + 10 points)**

a) 🡺 What is the hex value for these colours? **(15 points)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Red decimal** | **Green decimal** | **Blue decimal** | **Hex triplet** | **What  Colour?** |
| 192 | 255 | 238 | C0FFEE | Light Mint |
| 126 | 164 | 112 | 7EA470 | Dark Green |
| 186 | 187 | 30 | BABB1E | Olive |
| 208 | 13 | 30 | D00D1E | Dark Red |
| 15 | 245 | 231 | 0FF5E7 | Cyan |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **6 digit Hex code** | **Red decimal value (0-255)** | **Green decimal value (0-255)** | **Blue decimal value (0-255)** | **Describe the Final Colour *and* change the cell's background colour, i.e. R-click and see MS Word 'Shading', to match the values for RGB** |
| #D64A53 | 214 | 74 | 83 |  |
| #404893 | 64 | 72 | 147 |  |
| #302430 | 48 | 36 | 48 |  |
| #204C07 | 32 | 76 | 7 |  |