Patel vrundaben vijaykumar158605220 vvpatel20@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)

32-bit long = 2^32= 4,294,967,296 values

Maximum value = 256^4 **/** 2 **-**1= 2,147,483,647.

Hundreth of seconds of max. value = 21,474,836.47 seconds

Seconds to minutes= 21,474,836.47**/** 60 = 357913.9 minutes

Minutes to hours = 357913.9 **/** 60 = 5965.2 hours

Hours to days = 5965.2 **/** 24 **= 248.5 days** it will take until the integer 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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 249 | 5965 | 357914 | 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.

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

-17,569 + -17,000 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 calculation can go wrong if both low and high values are same data types, then the midle point value would overflow before the division takes place. When adding (low + high) first, the result of the middle point is higher than the capacity of particular memory that can store this numbers , due to this the addition can overflow and its behavior is undefined in c language.

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.

First I looked for the midbugtest.c file which is provided in the instruction and opened in the microsoft visual studio2022 and tried compiling it to find the exact problem which was with calculations because it was overflowing at first when addition (low+high)is being done before the division. To first find the solution, I first got any two values e.g 110 and 120, now to think logically I asked myself what will be the median of this values? And answer was 115. Now if I could add both the values 110+120 and if the result is greater than the memory can hold then it would cause overflow hence the behavior is undefined. So first I found the difference of the two values by doing 120-110 which is 10 and later I did 10 divide by 2 which is 5, then I added it to the lower value 110 which is now 115, and finally this is how I got the formula ,1st to get difference (high-low) then dividing by 2 and later adding the result to low value which is **low +((high-low) / 2 )**. Later I opened a new program in the visual studio and wrote a source code. I declared an array of size 10 : arr[10]=1,2,3,4,5,6,7,8,9,10, low, high, mid,n (the number to be checked in the array). After that, I used a for loop to iterate through every element of the array. Then I used formula mid = low + (high-low)/2 and set low to 0 and high to 9. then, I verified that low should be less than or equal to high While(low<=high). I checked the conditions if they satisfied or not if the n<mid then it would do high=mid-1, if the n>mid then it is low=mid+1. After building the program I found no errors in it and I debugged it and that’s how I got my calculation **low +((high-low) / 2 )**.

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 | Vivid cyan |
| 192 | 255 | 238 | #C0FFEE | Very pale cyan |
| 208 | 13 | 30 | #D00D1E | Strong Red |
| 186 | 187 | 30 | #BABB1E | Strong yellow |
| 126 | 164 | 112 | #7EA47A | Mostly desaturated dark lime green |

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 | Very dark (mostly black) magenta |
| #204C02 | 32 | 76 | 2 | Very dark green |
| #D64A53 | 214 | 73 | 84 | Moderate red |
| #404891 | 64 | 72 | 145 | Dark moderate blue |