Name: Cameron Peterson-Zopf

SMALL ASSIGNMENT WEEK 10

Discussion on a high level with your colleagues is encouraged. Make sure the work submitted is your own. When in doubt, ask a TA or the instructor. If you are not sure what constitutes academic dishonesty, please refer to the AISC web site: https://aisc.uci.edu/.

You can fill out your answers below in text, paste screenshots, and/or include images (make sure the image is right side up & legible).

Submit a single document showing/discussing the following:

- What are your valid input test cases & expected output for this sum function with 32-bit integers?
  - i.e. one test case was 5 w/ expected output 15
  - How many test cases do you have?
  - Show a screenshot of those test cases being run on the terminal (Linux or gdb is fine, Linux recommended)
- Do all 2^{32} inputs produce valid results?
  - If yes, show the input & results of the highest & lowest values.
  - If no,
    - show the input & results of the highest & lowest values that produce a valid results
    - show the input & results of the first values (highest/lowest) that produce invalid results
  - Include screenshots and explanation of your answers

AISC

Please initial here to indicate you understand UCI's Academic Integrity Policy and confirm that this is your own work you are submitting (this counts for points): CPZ

SCREENSHOT OF CODE:

The valid positive integers from just the default code are from 1 to 1,098,583. The final valid input value is 1,098,583 because this number will result in the final sum below 2^31 -1. The sum variable is of type integer and thus must be no greater than 2^31 -1. I have also added in the edge case of zero. All negative numbers are invalid as we are assuming positive numbers for this code. Thus, not all 2^32 inputs will produce valid results, only some positive ones and zero. All negative inputs will produce invalid results. Valid test cases are zero through 1,098,583, which is a total of 1,098,584 inputs.

The code program is stored in:

```
[chpeters@crystalcove ~/eecs20]$ ls -l
total 4
-rw-------. 1 chpeters ugrad 1296 Sep  5 17:30 debug_sum.c
[chpeters@crystalcove ~/eecs20]$
```

The code program contents are:

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

int AllSum(int n) {
    int result = 0;
    int i;  // counter variable, not used outside for loop, don't need initialize

    for (i = 1; i<=n; i++) {
        result = result + i;
    }
    return result; //sum of all numbers 1 through n.
}

// sums up all digits up to max digit user specifies
// example:
//   user input: 5
//   program output: 15 (basically: 1 + 2 + 3 + 4 + 5)
int main(void)
{
    printf("Welcome, this program calculates a sum from 1 to n, where n is an integer you specify.\n"); // welcome user
    printf("For example, an input of 5 will get an output of %d\n", AllSum(5));
    printf("Please input an integer from 0 to 2^31 - 1: ");
    int n;
    scanf("%d", &n); // get user input
    if (n >= 1)
    {
        int sum = AllSum(n); // sum it up
        printf("The sum is: %d", sum); // print summary
    }
    else if (n == 0)
    {
        printf("The sum is: 0");
    }
    else if (n < 0)
    {
        while (n < 0)
        {
            printf("Negative numbers are not allowed. Please input another integer: ");
            scanf("%d", &n);
        }
        int sum = AllSum(n); // sum it up
        printf("The sum is: %d", sum); // print summary
    }
        return 0;
}
/* EOF */
```

The file is compiled:

```
[chpeters@crystalcove ~/eecs20]$ gcc debug_sum.c -ansi -std=c99 -Wall -g
[chpeters@crystalcove ~/eecs20]$ ls -l
total 28
-rwx------. 1 chpeters ugrad 20960 Sep  5 17:34 a.out
-rw-------. 1 chpeters ugrad  1296 Sep  5 17:30 debug_sum.c
[chpeters@crystalcove ~/eecs20]$ gdb a.out
GNU gdb (GDB) Rocky Linux 8.2-20.el8.0.1
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from a.out...done.
(gdb)
```

TEST CASE: 0 (LOWEST VALID INPUT)

The first important test case is zero. The result should be zero.

```
(gdb) run
Starting program: /users/ugrad/chpeters/eecs20/a.out
Welcome, this program calculates a sum from 1 to n, where n is an integer you specify.
For example, an input of 5 will get an output of 15
Please input an integer from 0 to 2^31 - 1: 0
The sum is: 0[Inferior 1 (process 1433509) exited normally]
```

Also, the test case of 5 is shown at the top to give the user an example. The result of 15 shown is correct.

TEST CASE: 1,098,583 (HIGHEST VALID INPUT)

The second important test case is the last number we can test for an integer. Since the integer data type “int” is signed by default the max positive number is 2^31 -1, which is 2,147,483,647. This is what the max sum can be.

```
(gdb) run
Starting program: /users/ugrad/chpeters/eecs20/a.out
Welcome, this program calculates a sum from 1 to n, where n is an integer you specify.
For example, an input of 5 will get an output of 15
Please input an integer from 0 to 2^31 - 1: 1098583
The sum is: 2147431796[Inferior 1 (process 1433612) exited normally]
(gdb)
```

We see that the sum from 1,098,583 is 2,147,431,796. This value is slightly below 2^31 -1.

TEST CASE: NEGATIVE NUMBERS: FIRST NEGATIVE NUMBER IS -1

The third important test case is any negative number. Users cannot input negative numbers.

```
(gdb) run
Starting program: /users/ugrad/chpeters/eecs20/a.out
Welcome, this program calculates a sum from 1 to n, where n is an integer you specify.
For example, an input of 5 will get an output of 15
Please input an integer from 0 to 2^31 - 1: -1
Negative numbers are not allowed. Please input another integer: 10
The sum is: 55[Inferior 1 (process 1433680) exited normally]
(gdb)
```

After the input of -1 is stated as invalid, the user is prompted again where 10 is inputted.

TEST CASE: 1,098,584. THE FIRST POSITIVE NUMBER OUT OF RANGE

The fourth important test case is the first positive number for the input that will result in the sum variable being out of range, which is 1,098,584.

```
(gdb) run
Starting program: /users/ugrad/chpeters/eecs20/a.out
Welcome, this program calculates a sum from 1 to n, where n is an integer you specify.
For example, an input of 5 will get an output of 15
Please input an integer from 0 to 2^31 - 1: 1098584
The sum is: -2146436916[Inferior 1 (process 1433791) exited normally]
(gdb)
```

The answer is -2,146,436,916. The answer is negative because we have gone past the max positive range and now have wrapped back around to the negative side.