Cyber Assignment 2: Integer and String Calculator

Project Summary

October 28, 2025

1 Project Structure

Files:

- main.c, main.h - Main program loop and operation dispatcher
- parser.c, parser.h - Input parsing and validation
- overFlowChecker.c, overFlowChecker.h - Overflow detection
- add.c, sub.c, mult.c, div.c - Arithmetic operations
- Makefile - Build configuration
- tests.txt - Test cases

2 Code

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>
#include <ctype.h>
#include "./main.h"
#include "./parser.h"

void main()
{
    bool run = true;
    while (run) {
        // Read input
        char input[113];
        readSafeInput(input);
        int len = strlen(input);

        // exit case
        if(strcmp(input, "exit") == 0) {
            run = false;
            printf("%s\n", "Breaking out of Program...");
            break;
        }

        // Validate input characters first
        if (!validateInputCharacters(input)) {
            printf("Error: Input contains invalid characters. Only digits (0-9)
    , letters (a-z, A-Z), and operators (+, -, *, /, %%) are allowed.\n\n");
            continue;
        }
```

```
30
31          // get op Type
32          char opType = getOperationType(input);
33          if (opType == '0') {
34              printf("Error: No valid operator found in input.\n\n");
35              continue;
36          }
37
38          char left[101];
39          char right[101];
40          signed int num1;
41          signed int num2;
42          char charInput[101];
43          bool isStrOP = false;
44
45          parseTwoValues(input, left, right, opType);
46          int operationNum = getStrORNumOp(left, right, opType);
47
48          switch (operationNum) {
49              case 0:
50                  printf("%s\n\n", "Invalid Input! Ensure you have two numbers or
    a number and a string with a valid operation.");
51                  continue;
52              case 1:
53                  num1 = (signed int)atoi(left);
54                  num2 = (signed int)atoi(right);
55                  break;
56              case 2:
57                  num1 = (signed int)atoi(left);
58                  strcpy(charInput, right);
59                  isStrOP = true;
60                  break;
61              case 3:
62                  strcpy(charInput, left);
63                  num1 = (signed int)atoi(right);
64                  isStrOP = true;
65                  break;
66          }
67
68          signed int ans;
69          char multCharOutput[1025];  // 1024 + null terminator
70          bool errorOperating = false;
71          bool isMult = false;
72
73          switch (opType) {
74              case '+':
75                  printf("%s\n", "Operation type: +");
76                  if (isStrOP) {
77                      addModString(num1, strlen(charInput), charInput);
78                      break;
79                  }
80                  if (!addNum(num1, num2, &ans)) {
81                      errorOperating = true;
82                      printf("%s\n", "Error adding numbers; integer overflow
   detected.");
83                  }
84                  break;
85              case '-':
86                  printf("%s\n", "Operation type: -");
87                  if (isStrOP) {
88                      subModString(num1, strlen(charInput), charInput);
89                      break;
90                  }
```

```
                if (!subNum(num1, num2, &ans)) {
                    errorOperating = true;
                    printf("%s\n", "Error subtracting numbers; integer overflow
    /underflow detected.");
                }
                break;
            case '/':
                printf("%s\n", "Operation type: /");
                if (isStrOP) {
                    divModString(num1, strlen(charInput), charInput);
                    break;
                }
                if (!divNum(num1, num2, &ans)) {
                    errorOperating = true;
                    printf("%s\n", "Error dividing numbers; division by zero.")
    ;
                }
                break;
            case '*':
                isMult = true;
                printf("%s\n", "Operation type: *");
                if (isStrOP) {
                    multModString(num1, strlen(charInput), charInput,
    multCharOutput);
                    break;
                }
                if (!multNum(num1, num2, &ans)) {
                    errorOperating = true;
                    printf("%s\n", "Error multiplying numbers; integer overflow
     detected.");
                }
                break;
            case '%':
                printf("%s\n", "Operation type: %");
                if (!modNum(num1, num2, &ans)) {
                    errorOperating = true;
                    printf("%s\n", "Error modding numbers; modulo by zero.");
                }
                break;
        }
        if (!errorOperating) {
            if (isStrOP) {
                if (isMult) {
                    printf("%s %s\n\n", "Output String is:", multCharOutput);
                } else {
                    printf("%s %s\n\n", "Output String is:", charInput);
                }
            } else {
                printf("%s %d\n\n", "Output Number is:", ans);
            }
        } else {
            printf("\n");
        }
    }
};
```

Listing 1: main.c

2.1 main.h

```
#include <stdio.h>
#include <stdbool.h>
```

```
3
4  bool addNum(signed int Num1, signed int Num2, signed int *ans);
5  void addModString(int Num, int len, char InputString[]);
6
7  bool subNum(signed int Num1, signed int Num2, signed int* ans);
8  void subModString(int Num, int len, char InputString[]);
9
10 bool multNum(signed int Num1, signed int Num2, signed int* ans);
11 void multModString(int Num, int len, char InputString[], char multCharOutput[])
       ;
12
13 bool divNum(signed int Num1, signed int Num2, signed int* ans);
14 bool modNum(signed int Num1, signed int Num2, signed int* ans);
15 void divModString(int Num, int len, char InputString[]);
```

Listing 2: main.h

2.2 parser.c

```
1  #include <stdio.h>
2  #include <stdbool.h>
3  #include <string.h>
4  #include <stdlib.h>
5  #include <limits.h>
6
7  // 0 is invalid, 1 is +, 2 is -, 3 is /, 4 is *, 5 is %
8  char getOperationType(char input[])
9  {
10     if (input == NULL) return 0;
11     for (int i = 0; input[i] != '\0'; ++i) {
12         char c = input[i];
13         switch (c) {
14             case '+': return c;
15             case '-': return c;
16             case '/': return c;
17             case '*': return c;
18             case '%': return c;
19             default: break;
20         }
21     }
22     return '0';
23 }
24
25 // Validate that input contains only allowed characters
26 bool validateInputCharacters(char input[]) {
27     for (int i = 0; input[i] != '\0'; i++) {
28         char c = input[i];
29         // Allow: digits, letters (a-z, A-Z), and operators (+, -, *, /, %)
30         if (!((c >= '0' && c <= '9') ||
31               (c >= 'a' && c <= 'z') ||
32               (c >= 'A' && c <= 'Z') ||
33               c == '+' || c == '-' || c == '*' || c == '/' || c == '%')) {
34             return false;
35         }
36     }
37     return true;
38 }
39
40 // reads input.
41 void readSafeInput(char input[])
42 {
43     // 100 chars and a int up to int max, 11 chars, plus a operator 1 char,
       plus null terminator, buffer 113 chars
```

```
    char buff[113];
    int max = sizeof(buff);
    printf("Please enter (1) two integers or (2) one integer and one string (up to\n");
    printf("100 characters, contains letters a-z and/or A-Z only). Enter everything in one\n");
    printf("line, separated by an operator (NO WHITESPACES): \n");

    if (fgets(buff, max, stdin) != NULL) {
        // Remove trailing newline if present
        size_t len = strlen(buff);
        if (len > 0 && buff[len-1] == '\n') {
            buff[len-1] = '\0';
        }
        // Copy to output parameter
        strcpy(input, buff);
    }
};

// splits input into two values
void parseTwoValues(char input[], char out1[], char out2[], char opType)
{
    int out1d_index = 0;
    int out2d_index = 0;
    bool firstDone = false;

    for (int i = 0; i < strlen(input); i++) {
        if (!firstDone) {
            if (input[i] == opType) {
                firstDone = true;
                continue;
            }
            out1[out1d_index++] = input[i];
        } else {
            out2[out2d_index++] = input[i];
        }
    }
    out1[out1d_index] = '\0';
    out2[out2d_index] = '\0';
};
```

Listing 3: parser.c (Part 1/2)

```
// Validate that string contains only letters a-z and A-Z
bool isValidString(char str[]) {
    for (int i = 0; str[i] != '\0'; i++) {
        if (!((str[i] >= 'a' && str[i] <= 'z') || (str[i] >= 'A' && str[i] <= 'Z'))) {
            return false;
        }
    }
    return true;
}

// Validate that string contains only digits
bool isValidNumber(char str[]) {
    for (int i = 0; str[i] != '\0'; i++) {
        if (str[i] < '0' || str[i] > '9') {
            return false;
        }
    }
    return true;
}

```

```
// Check if number string is in valid range [0, INT_MAX]
bool isNumberInRange(char str[]) {
    if (strlen(str) == 0) return false;

    char* endptr;
    long long val = strtoll(str, &endptr, 10);

    if (*endptr != '\0') return false;

    if (val < 0 || val > INT_MAX) {
        return false;
    }

    return true;
}

// 0 is invalid, 1 is two nums, 2 is num and char, 3 is char and num
int getStrORNumOp(char out1[], char out2[], char opType)
{
    bool out1isNum = isValidNumber(out1);
    bool out2isNum = isValidNumber(out2);

    if (strlen(out1) == 0 || strlen(out2) == 0) {
        printf("Error: Empty operand detected.\n");
        return 0;
    }

    if (out1isNum && out2isNum) {
        if (!isNumberInRange(out1)) {
            printf("Error: First number is out of range [0, %d].\n", INT_MAX);
            return 0;
        }
        if (!isNumberInRange(out2)) {
            printf("Error: Second number is out of range [0, %d].\n", INT_MAX);
            return 0;
        }
        return 1;  // two nums
    }

    if (opType == '%') {
        printf("Error: Modulo operation (%%) requires two integers.\n");
        return 0;
    }

    if (!out1isNum && !out2isNum) {
        printf("Error: Two strings entered as operands. Only accept two
    integers or one integer and one string.\n");
        return 0;
    }

    if (out1isNum) {
        if (!isNumberInRange(out1)) {
            printf("Error: Number is out of range [0, %d].\n", INT_MAX);
            return 0;
        }
        if (strlen(out2) > 100) {
            printf("Error: String has more than 100 characters.\n");
            return 0;
        }
        if (!isValidString(out2)) {
            printf("Error: String contains invalid characters. Only a-z and A-Z
     allowed.\n");
            return 0;
```

```
        }
        return 2;  // num and char
    } else {
        if (!isNumberInRange(out2)) {
            printf("Error: Number is out of range [0, %d].\n", INT_MAX);
            return 0;
        }
        if (strlen(out1) > 100) {
            printf("Error: String has more than 100 characters.\n");
            return 0;
        }
        if (!isValidString(out1)) {
            printf("Error: String contains invalid characters. Only a-z and A-Z allowed.\n");
            return 0;
        }
        return 3;  // char and num
    }
}
```

Listing 4: parser.c (Part 2/2)

2.3 parser.h

```
#include <stdio.h>
#include <stdbool.h>

char getOperationType(char input[]); // 0 is invalid, +, -, /, *, %
void readSafeInput(char input[]); // reads input.
bool parseTwoValues(char input[], char out1[], char out2[], char opType); // splits input into two values
int getStrORNumOp(char out1[], char out2[], char opType); // 0 is invalid, 1 is two nums, 2 is num and char, 3 is char and num
bool validateInputCharacters(char input[]);
```

Listing 5: parser.h

2.4 overFlowChecker.c

```
#include <stdio.h>
#include <stdbool.h>
#include <limits.h>
#include "./overFlowChecker.h"

bool isValidAddOp(int Num1, int Num2) {
  if (Num2 > 0 && Num1 > INT_MAX - Num2) return false;
  if (Num2 < 0 && Num1 < INT_MIN - Num2) return false;
  return true;
};

bool isValidSubOp(int Num1, int Num2)
{
  if (Num2 < 0 && Num1 > INT_MAX + Num2) return false;
  if (Num2 > 0 && Num1 < INT_MIN + Num2) return false;
  return true;
};

bool isValidMultOp(int Num1, int Num2)
{
  if (Num1 == 0 || Num2 == 0) return true;
  long long prod = (long long)Num1 * (long long)Num2;
```

```
  if (prod > INT_MAX || prod < INT_MIN) return false;
  return true;
};

bool isValidDivOrModOp(int Num1, int Num2)
{
  if (Num2 == 0) return false;
  if (Num1 == INT_MIN && Num2 == -1) return false;
  return true;
};
```

Listing 6: overFlowChecker.c

2.5 overFlowChecker.h

```
#include <stdio.h>
#include <stdbool.h>

bool isValidAddOp(int Num1, int Num2);
bool isValidSubOp(int Num1, int Num2);
bool isValidMultOp(int Num1, int Num2);
bool isValidDivOrModOp(int Num1, int Num2);
```

Listing 7: overFlowChecker.h

2.6 add.c

```
#include <stdio.h>
#include "./overFlowChecker.h"
#include "./main.h"

bool addNum(int Num1, int Num2, int* ans)
{
    if (!isValidAddOp(Num1, Num2)) return false;
    *ans = Num1+Num2;
    return true;
};

void addModString(int Num, int len, char InputString[])
{
    // Shift each letter right by Num positions in the alphabet
    int shift = Num % 26;  // Only need to shift by remainder when divided by
    26

    for (int i = 0; i < len; i++)
    {
        if (InputString[i] >= 'a' && InputString[i] <= 'z') {
            // Lowercase letter
            InputString[i] = 'a' + (InputString[i] - 'a' + shift) % 26;
        } else if (InputString[i] >= 'A' && InputString[i] <= 'Z') {
            // Uppercase letter
            InputString[i] = 'A' + (InputString[i] - 'A' + shift) % 26;
        }
    }
};
```

Listing 8: add.c

2.7 sub.c

```c
#include <stdio.h>
#include "./overFlowChecker.h"
#include "./main.h"

bool subNum(int Num1, int Num2, int* ans)
{
    if (!isValidSubOp(Num1, Num2)) return false;
    *ans = Num1-Num2;
    return true;
};

void subModString(int Num, int len, char InputString[])
{
    // Shift each letter left by Num positions in the alphabet
    int shift = Num % 26;  // Only need to shift by remainder when divided by 26

    for (int i = 0; i < len; i++)
    {
        if (InputString[i] >= 'a' && InputString[i] <= 'z') {
            // Lowercase letter
            InputString[i] = 'a' + (InputString[i] - 'a' - shift + 26) % 26;
        } else if (InputString[i] >= 'A' && InputString[i] <= 'Z') {
            // Uppercase letter
            InputString[i] = 'A' + (InputString[i] - 'A' - shift + 26) % 26;
        }
    }
};
```

Listing 9: sub.c

2.8 mult.c

```c
#include <stdio.h>
#include <string.h>
#include "./overFlowChecker.h"
#include "./main.h"

bool multNum(int Num1, int Num2, int* ans)
{
    if (!isValidMultOp(Num1, Num2)) return false;
    *ans = Num1*Num2;
    return true;
};

void multModString(int Num, int len, char InputString[], char multCharOutput[])
{
    multCharOutput[0] = '\0';  // Initialize empty string

    for (int i = 0; i < Num; i++) {
        // Check if adding another copy would exceed 1024 characters
        if (strlen(multCharOutput) + len > 1024) {
            // Only add as much as possible to reach 1024
            int remaining = 1024 - strlen(multCharOutput);
            strncat(multCharOutput, InputString, remaining);
            printf("Warning: Result string is too long. Truncated to 1024 characters.\n");
            break;
        }
        strcat(multCharOutput, InputString);
    }
```

```
};
```

Listing 10: mult.c

2.9 div.c

```c
#include <stdio.h>
#include "./overFlowChecker.h"
#include "./main.h"

bool divNum(int Num1, int Num2, int* ans)
{
    if (!isValidDivOrModOp(Num1, Num2)) return false;
    *ans = Num1/Num2;
    return true;
};

bool modNum(int Num1, int Num2, int* ans)
{
    if (!isValidDivOrModOp(Num1, Num2)) return false;
    *ans = Num1%Num2;
    return true;
};

void divModString(int Num, int len, char InputString[])
{
    // Cut string from the end by Num characters
    if (Num >= len) {
        // If Num is larger than or equal to string length, result is empty
    string
        InputString[0] = '\0';
    } else {
        // Terminate the string at the new length
        InputString[len - Num] = '\0';
    }
};
```

Listing 11: div.c

2.10 Makefile

```
# Simple Makefile for calc project
# To build: type "make"
# To clean: type "make clean"

# Build the calc program
calc: main.c add.c sub.c mult.c div.c parser.c overFlowChecker.c
  gcc -o calc main.c add.c sub.c mult.c div.c parser.c overFlowChecker.c

# Remove the compiled program
clean:
  rm -f calc
```

Listing 12: Makefile

3 Test Results

```
Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
```

```
line, separated by an operator (NO WHITESPACES):
220+cyb
Operation type: +
Output String is: okn

Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
CYBERsecurity-5
Operation type: -
Output String is: XTWZMnzxpmdot

Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
6*zzzz
Operation type: *
Output String is: zzzzzzzzzzzzzzzzzzzzzzz

Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
CYB*220
Operation type: *
Output String is:
    CYBCYBCYBCYBCYBCYBCYBCYBCYBCYBCYBCYBCYBCYBCYBCYBCYBCYBCYBCYBCYBCYBCYBCYBCYBCYBCYBCYBCY


Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
Cybersecurity/8
Operation type: /
Output String is: Cyber

Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
20/Cybersecurity
Operation type: /
Output String is:

Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
100%7
Operation type: %
Output Number is: 2

Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
2000000000+300000000
Operation type: +
Error adding numbers; integer overflow detected.

Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
123+4000000000
Error: Second number is out of range [0, 2147483647].
Invalid Input! Ensure you have two numbers or a number and a string with a
    valid operation.
```

```

Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
77777*100000
Operation type: *
Error multiplying numbers; integer overflow detected.

Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
-acb123
Error: Empty operand detected.
Invalid Input! Ensure you have two numbers or a number and a string with a
    valid operation.

Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
abcde*500
Operation type: *
Warning: Result string is too long. Truncated to 1024 characters.
Output String is:
    abcdeabcdeabcdeabcdeabcdeabcdeabcdeabcdeabcdeabcdeabcdeabcdeabcdeabcdeabcdeabcdeabcdeabcd


Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
9+abc;def
Error: Input contains invalid characters. Only digits (0-9), letters (a-z, A-Z)
    , and operators (+, -, *, /, %) are allowed.

Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
123+5+6+7
Error: String contains invalid characters. Only a-z and A-Z allowed.
Invalid Input! Ensure you have two numbers or a number and a string with a
    valid operation.

Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
Abc-xyz
Error: Two strings entered as operands. Only accept two integers or one integer
     and one string.
Invalid Input! Ensure you have two numbers or a number and a string with a
    valid operation.

Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
8000000000+1
Error: First number is out of range [0, 2147483647].
Invalid Input! Ensure you have two numbers or a number and a string with a
    valid operation.

Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
3*
    aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

```
Error: String has more than 100 characters.
Invalid Input! Ensure you have two numbers or a number and a string with a valid operation.

Please enter (1) two integers or (2) one integer and one string (up to 100 characters, contains letters a-z and/or A-Z only). Enter everything in one line, separated by an operator (NO WHITESPACES):

Error: No valid operator found in input.

Please enter (1) two integers or (2) one integer and one string (up to 100 characters, contains letters a-z and/or A-Z only). Enter everything in one line, separated by an operator (NO WHITESPACES):
500%0
Operation type: %
Error modding numbers; modulo by zero.

Please enter (1) two integers or (2) one integer and one string (up to 100 characters, contains letters a-z and/or A-Z only). Enter everything in one line, separated by an operator (NO WHITESPACES):
Abc%4
Error: Modulo operation (%) requires two integers.
Invalid Input! Ensure you have two numbers or a number and a string with a valid operation.

Please enter (1) two integers or (2) one integer and one string (up to 100 characters, contains letters a-z and/or A-Z only). Enter everything in one line, separated by an operator (NO WHITESPACES):
3 + 0
Error: Input contains invalid characters. Only digits (0-9), letters (a-z, A-Z), and operators (+, -, *, /, %) are allowed.

Please enter (1) two integers or (2) one integer and one string (up to 100 characters, contains letters a-z and/or A-Z only). Enter everything in one line, separated by an operator (NO WHITESPACES):
abc+1
Operation type: +
Output String is: bcd

Please enter (1) two integers or (2) one integer and one string (up to 100 characters, contains letters a-z and/or A-Z only). Enter everything in one line, separated by an operator (NO WHITESPACES):
abc+26
Operation type: +
Output String is: abc

Please enter (1) two integers or (2) one integer and one string (up to 100 characters, contains letters a-z and/or A-Z only). Enter everything in one line, separated by an operator (NO WHITESPACES):
ABC+25
Operation type: +
Output String is: ZAB

Please enter (1) two integers or (2) one integer and one string (up to 100 characters, contains letters a-z and/or A-Z only). Enter everything in one line, separated by an operator (NO WHITESPACES):
abc-1
Operation type: -
Output String is: zab

Please enter (1) two integers or (2) one integer and one string (up to 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
```

```
line, separated by an operator (NO WHITESPACES):
abcd*2
Operation type: *
Output String is: abcdabcd

Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
homework/3
Operation type: /
Output String is: homew

Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
homework/8
Operation type: /
Output String is:

Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
10+20
Operation type: +
Output Number is: 30

Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
100-50
Operation type: -
Output Number is: 50

Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
5*6
Operation type: *
Output Number is: 30

Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
20/4
Operation type: /
Output Number is: 5

Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
2147483647+0
Operation type: +
Output Number is: 2147483647

Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
2147483647+1
Operation type: +
Error adding numbers; integer overflow detected.

Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
```

```
line, separated by an operator (NO WHITESPACES):
10/0
Operation type: /
Error dividing numbers; division by zero.

Please enter (1) two integers or (2) one integer and one string (up to
100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
exit
Breaking out of Program...
```

Listing 13: tests.txt