# 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

```
#include <stdio.h>
#include <string.h>
3 #include <stdlib.h>
4 #include <stdbool.h>
5 #include <ctype.h>
6 #include "./main.h"
7 #include "./parser.h"
9 void main()
10 {
      bool run = true;
      while (run) {
12
          // Read input
13
          char input[113];
14
          readSafeInput(input);
          int len = strlen(input);
          // exit case
          if(strcmp(input, "exit") == 0) {
19
              run = false;
20
              printf("%s\n", "Breaking out of Program...");
22
              break;
23
24
          // 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;
28
```

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

```
if (!subNum(num1, num2, &ans)) {
91
                        errorOperating = true;
92
                        93
       /underflow detected.");
94
                   break;
95
               case '/':
96
97
                   printf("%s\n", "Operation type: /");
                    if (isStrOP) {
                        divModString(num1, strlen(charInput), charInput);
99
100
                   }
                   if (!divNum(num1, num2, &ans)) {
                        errorOperating = true;
                        printf("%s\n", "Error dividing numbers; division by zero.")
                   }
106
                   break;
               case '*':
107
                   isMult = true;
108
                   printf("%s\n", "Operation type: *");
109
110
                   if (isStrOP) {
111
                        multModString(num1, strlen(charInput), charInput,
      multCharOutput);
                       break;
113
                    if (!multNum(num1, num2, &ans)) {
114
                       errorOperating = true;
printf("%s\n", "Error multiplying numbers; integer overflow
116
       detected.");
                   break;
               case '%':
119
                   printf("%s\n", "Operation type: %");
120
                   if (!modNum(num1, num2, &ans)) {
                        errorOperating = true;
                       printf("%s\n", "Error modding numbers; modulo by zero.");
123
                   }
124
                   break;
125
126
           if (!errorOperating) {
127
128
               if (isStrOP) {
                   if (isMult) {
129
                       printf("%s %s\n\n", "Output String is:", multCharOutput);
130
                   } else {
131
                       printf("%s %s\n\n", "Output String is:", charInput);
132
133
134
               } else {
                   printf("%s %d\n\n", "Output Number is:", ans);
136
           } else {
137
               printf("\n");
138
           }
139
       }
140
141 };
```

Listing 1: main.c

### 2.1 main.h

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

```
bool addNum(signed int Num1, signed int Num2, signed int *ans);
void addModString(int Num, int len, char InputString[]);

bool subNum(signed int Num1, signed int Num2, signed int* ans);
void subModString(int Num, int len, char InputString[]);

bool multNum(signed int Num1, signed int Num2, signed int* ans);
void multModString(int Num, int len, char InputString[], char multCharOutput[])
;

bool divNum(signed int Num1, signed int Num2, signed int* ans);
bool modNum(signed int Num1, signed int Num2, signed int* ans);
void divModString(int Num, int len, char InputString[]);
```

Listing 2: main.h

### 2.2 parser.c

```
#include <stdio.h>
#include <stdbool.h>
3 #include <string.h>
4 #include <stdlib.h>
5 #include <limits.h>
_{7} // 0 is invalid, 1 is +, 2 is -, 3 is /, 4 is *, 5 is %
8 char getOperationType(char input[])
9 {
      if (input == NULL) return 0;
10
      for (int i = 0; input[i] != '\0'; ++i) {
          char c = input[i];
          switch (c) {
              case '+': return c;
14
               case '-': return c;
               case '/': return c;
16
17
               case '*': return c;
18
               case '%': return c;
19
               default: break;
          }
20
      }
21
      return '0';
22
23 }
24
25 // Validate that input contains only allowed characters
26 bool validateInputCharacters(char input[]) {
      for (int i = 0; input[i] != '\0'; i++) {
27
           char c = input[i];
           // Allow: digits, letters (a-z, A-Z), and operators (+, -, *, /, %)
          if (!((c >= '0' && c <= '9') ||
                 (c >= 'a' && c <= 'z') ||
31
                 (c >= 'A' && c <= 'Z') ||
32
                 c == '+' || c == '-' || c == '*' || c == '/' || c == '%')) {
33
34
               return false;
          }
35
      }
36
37
      return true;
38 }
40 // reads input.
41 void readSafeInput(char input[])
42 {
      // 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];
44
      int max = sizeof(buff);
45
      printf("Please enter (1) two integers or (2) one integer and one string (up
46
       to\n");
      printf("100 characters, contains letters a-z and/or A-Z only). Enter
47
      everything in one\n");
      printf("line, separated by an operator (NO WHITESPACES): \n");
48
       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] = ' \setminus 0';
54
           // Copy to output parameter
56
           strcpy(input, buff);
57
58
59 };
61 // splits input into two values
62 void parseTwoValues(char input[], char out1[], char out2[], char opType)
63 {
64
       int out1d_index = 0;
      int out2d_index = 0;
65
      bool firstDone = false;
66
67
      for (int i = 0; i < strlen(input); i++) {</pre>
68
           if (!firstDone) {
69
               if (input[i] == opType) {
70
                    firstDone = true;
72
                    continue;
               }
73
               out1[out1d_index++] = input[i];
74
           } else {
75
               out2[out2d_index++] = input[i];
76
77
78
      out1[out1d_index] = '\0';
79
      out2[out2d_index] = '\0';
80
81 };
```

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

```
_{1} // Validate that string contains only letters a-z and A-Z
2 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;
          }
7
      }
8
      return true;
9 }
10
11 // Validate that string contains only digits
12 bool isValidNumber(char str[]) {
      for (int i = 0; str[i] != '\0'; i++) {
13
           if (str[i] < '0' || str[i] > '9') {
14
               return false;
          }
16
      }
17
18
      return true;
19 }
20
```

```
21 // Check if number string is in valid range [0, INT_MAX]
22 bool isNumberInRange(char str[]) {
      if (strlen(str) == 0) return false;
24
25
      char* endptr;
26
      long long val = strtoll(str, &endptr, 10);
27
      if (*endptr != '\0') return false;
      if (val < 0 || val > INT_MAX) {
30
31
          return false;
32
33
      return true;
34
35 }
_{
m 37} // O 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);
40
41
      bool out2isNum = isValidNumber(out2);
42
      if (strlen(out1) == 0 || strlen(out2) == 0) {
43
           printf("Error: Empty operand detected.\n");
44
           return 0;
45
46
47
      if (out1isNum && out2isNum) {
48
           if (!isNumberInRange(out1)) {
49
               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);
54
               return 0;
55
          }
56
          return 1; // two nums
57
58
      if (opType == '%') {
60
           printf("Error: Modulo operation (%%) requires two integers.\n");
          return 0;
62
      }
63
64
      if (!out1isNum && !out2isNum) {
65
          printf("Error: Two strings entered as operands. Only accept two
66
      integers or one integer and one string.\n");
67
          return 0;
68
      if (out1isNum) {
          if (!isNumberInRange(out1)) {
               printf("Error: Number is out of range [0, %d].\n", INT_MAX);
72
73
               return 0;
          }
74
          if (strlen(out2) > 100) {
75
               printf("Error: String has more than 100 characters.\n");
76
               return 0;
77
78
79
           if (!isValidString(out2)) {
              printf("Error: String contains invalid characters. Only a-z and A-Z
       allowed.\n");
           return 0;
```

```
}
82
          return 2; // num and char
83
      } else {
84
          if (!isNumberInRange(out2)) {
85
              printf("Error: Number is out of range [0, %d].\n", INT_MAX);
86
               return 0;
87
88
           if (strlen(out1) > 100) {
               printf("Error: String has more than 100 characters.\n");
               return 0;
          }
           if (!isValidString(out1)) {
93
               printf("Error: String contains invalid characters. Only a-z and A-Z
94
       allowed.\n");
              return 0;
95
96
          return 3; // char and num
97
99 }
```

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

## 2.3 parser.h

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

that 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>
3 #include <limits.h>
# #include "./overFlowChecker.h"
6 bool isValidAddOp(int Num1, int Num2) {
   if (Num2 > 0 && Num1 > INT_MAX - Num2) return false;
    if (Num2 < 0 && Num1 < INT_MIN - Num2) return false;</pre>
   return true;
10 };
bool isValidSubOp(int Num1, int Num2)
   if (Num2 < 0 && Num1 > INT_MAX + Num2) return false;
   if (Num2 > 0 && Num1 < INT_MIN + Num2) return false;</pre>
   return true;
17 };
18
19 bool isValidMultOp(int Num1, int Num2)
20 €
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;
};</pre>
```

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>
2 #include "./overFlowChecker.h"
3 #include "./main.h"
5 bool addNum(int Num1, int Num2, int* ans)
      if (!isValidAddOp(Num1, Num2)) return false;
      *ans = Num1+Num2;
9
      return true;
10 };
void addModString(int Num, int len, char InputString[])
13 {
      // Shift each letter right by Num positions in the alphabet
14
      int shift = Num % 26; // Only need to shift by remainder when divided by
      26
      for (int i = 0; i < len; i++)</pre>
17
18
           if (InputString[i] >= 'a' && InputString[i] <= 'z') {</pre>
19
               // Lowercase letter
2.0
               InputString[i] = 'a' + (InputString[i] - 'a' + shift) % 26;
2.1
           } else if (InputString[i] >= 'A' && InputString[i] <= 'Z') {</pre>
22
               // Uppercase letter
23
               InputString[i] = 'A' + (InputString[i] - 'A' + shift) % 26;
24
          }
25
      }
27 };
```

Listing 8: add.c

#### 2.7 sub.c

```
#include <stdio.h>
2 #include "./overFlowChecker.h"
3 #include "./main.h"
5 bool subNum(int Num1, int Num2, int* ans)
6 {
      if (!isValidSubOp(Num1, Num2)) return false;
      *ans = Num1-Num2;
9
      return true;
10 };
void subModString(int Num, int len, char InputString[])
      // Shift each letter left by Num positions in the alphabet
14
      int shift = Num % 26; // Only need to shift by remainder when divided by
16
17
      for (int i = 0; i < len; i++)</pre>
18
           if (InputString[i] >= 'a' && InputString[i] <= 'z') {</pre>
               // Lowercase letter
20
21
               InputString[i] = 'a' + (InputString[i] - 'a' - shift + 26) % 26;
22
          } else if (InputString[i] >= 'A' && InputString[i] <= 'Z') {</pre>
23
               // Uppercase letter
               InputString[i] = 'A' + (InputString[i] - 'A' - shift + 26) % 26;
24
          }
25
26
      }
27 };
```

Listing 9: sub.c

### 2.8 mult.c

```
#include <stdio.h>
#include <string.h>
3 #include "./overFlowChecker.h"
4 #include "./main.h"
6 bool multNum(int Num1, int Num2, int* ans)
7 {
      if (!isValidMultOp(Num1, Num2)) return false;
      *ans = Num1*Num2;
9
      return true;
10
11 };
12
13 void multModString(int Num, int len, char InputString[], char multCharOutput[])
14 {
      multCharOutput[0] = '\0'; // Initialize empty string
15
16
      for (int i = 0; i < Num; i++) {</pre>
17
          // Check if adding another copy would exceed 1024 characters
18
          if (strlen(multCharOutput) + len > 1024) {
19
               // Only add as much as possible to reach 1024
20
               int remaining = 1024 - strlen(multCharOutput);
21
               strncat(multCharOutput, InputString, remaining);
22
              printf("Warning: Result string is too long. Truncated to 1024
23
      characters.\n");
              break;
24
          strcat(multCharOutput, InputString);
26
```

28 };

Listing 10: mult.c

#### 2.9 div.c

```
#include <stdio.h>
#include "./overFlowChecker.h"
3 #include "./main.h"
5 bool divNum(int Num1, int Num2, int* ans)
6 {
      if (!isValidDivOrModOp(Num1, Num2)) return false;
      *ans = Num1/Num2;
8
9
      return true;
10 };
12 bool modNum(int Num1, int Num2, int* ans)
13 {
      if (!isValidDivOrModOp(Num1, Num2)) return false;
14
      *ans = Num1%Num2;
15
      return true;
16
17 };
18
19 void divModString(int Num, int len, char InputString[])
21
      // Cut string from the end by Num characters
22
      if (Num >= len) {
          // If Num is larger than or equal to string length, result is empty
23
      string
          InputString[0] = '\0';
24
      } else {
25
          // Terminate the string at the new length
26
          InputString[len - Num] = '\0';
27
28
29 };
```

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 | rm -f calc
```

Listing 12: Makefile

# 3 Test Results

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

```
3 line, separated by an operator (NO WHITESPACES):
4 220+cyb
5 Operation type: +
6 Output String is: okn
{\it 8} Please enter (1) two integers or (2) one integer and one string (up to
9 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
10 line, separated by an operator (NO WHITESPACES):
11 CYBERsecurity-5
12 Operation type: -
13 Output String is: XTWZMnzxpmdot
_{15} Please enter (1) two integers or (2) one integer and one string (up to
16 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
17 line, separated by an operator (NO WHITESPACES):
18 6*zzzz
19 Operation type: *
20 Output String is: zzzzzzzzzzzzzzzzzzzzzzz
22 Please enter (1) two integers or (2) one integer and one string (up to
23 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
24 line, separated by an operator (NO WHITESPACES):
25 CYB * 220
26 Operation type: *
27 Output String is:
      29 Please enter (1) two integers or (2) one integer and one string (up to
30 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
31 line, separated by an operator (NO WHITESPACES):
32 Cybersecurity/8
33 Operation type: /
34 Output String is: Cyber
_{36} Please enter (1) two integers or (2) one integer and one string (up to
37 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
38 line, separated by an operator (NO WHITESPACES):
39 20/Cybersecurity
40 Operation type: /
41 Output String is:
^{43} Please enter (1) two integers or (2) one integer and one string (up to
^{44} 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
45 line, separated by an operator (NO WHITESPACES):
46 100%7
47 Operation type: %
48 Output Number is: 2
50 Please enter (1) two integers or (2) one integer and one string (up to
51 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
52 line, separated by an operator (NO WHITESPACES):
53 200000000+30000000
54 Operation type: +
55 Error adding numbers; integer overflow detected.
57 Please enter (1) two integers or (2) one integer and one string (up to
58 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
59 line, separated by an operator (NO WHITESPACES):
60 123+4000000000
61 Error: Second number is out of range [0, 2147483647].
62 Invalid Input! Ensure you have two numbers or a number and a string with a
  valid operation.
```

```
63
_{64} Please enter (1) two integers or (2) one integer and one string (up to
_{65} 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
^{66} line, separated by an operator (NO WHITESPACES):
67 77777*100000
68 Operation type: *
69 Error multiplying numbers; integer overflow detected.
_{71} Please enter (1) two integers or (2) one integer and one string (up to
72 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
73 line, separated by an operator (NO WHITESPACES):
74 -acb123
75 Error: Empty operand detected.
76 Invalid Input! Ensure you have two numbers or a number and a string with a
      valid operation.
78 Please enter (1) two integers or (2) one integer and one string (up to
79 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
80 line, separated by an operator (NO WHITESPACES):
81 abcde *500
82 Operation type: *
83 Warning: Result string is too long. Truncated to 1024 characters.
84 Output String is:
      86 Please enter (1) two integers or (2) one integer and one string (up to
87 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
88 line, separated by an operator (NO WHITESPACES):
89 9+abc; def
90 Error: Input contains invalid characters. Only digits (0-9), letters (a-z, A-Z)
      , and operators (+, -, *, /, %) are allowed.
92 Please enter (1) two integers or (2) one integer and one string (up to
93 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
94 line, separated by an operator (NO WHITESPACES):
95 123+5+6+7
96 Error: String contains invalid characters. Only a-z and A-Z allowed.
97 Invalid Input! Ensure you have two numbers or a number and a string with a
      valid operation.
99 Please enter (1) two integers or (2) one integer and one string (up to
_{100} 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
102 Abc-xyz
\scriptstyle 103 Error: Two strings entered as operands. Only accept two integers or one integer
      and one string.
104 Invalid Input! Ensure you have two numbers or a number and a string with a
     valid operation.
_{106} Please enter (1) two integers or (2) one integer and one string (up to
107 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
108 line, separated by an operator (NO WHITESPACES):
109 8000000000+1
110 Error: First number is out of range [0, 2147483647].
111 Invalid Input! Ensure you have two numbers or a number and a string with a
      valid operation.
113 Please enter (1) two integers or (2) one integer and one string (up to
114 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
```

```
117 Error: String has more than 100 characters.
118 Invalid Input! Ensure you have two numbers or a number and a string with a
      valid operation.
119
_{120} Please enter (1) two integers or (2) one integer and one string (up to
121 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
122 line, separated by an operator (NO WHITESPACES):
124 Error: No valid operator found in input.
_{126} Please enter (1) two integers or (2) one integer and one string (up to
_{127} 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
128 line, separated by an operator (NO WHITESPACES):
129 500%0
130 Operation type: %
131 Error modding numbers; modulo by zero.
133 Please enter (1) two integers or (2) one integer and one string (up to
134 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
135 line, separated by an operator (NO WHITESPACES):
136 Abc %4
137 Error: Modulo operation (%) requires two integers.
138 Invalid Input! Ensure you have two numbers or a number and a string with a
      valid operation.
139
_{
m 140} Please enter (1) two integers or (2) one integer and one string (up to
_{141} 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
142 line, separated by an operator (NO WHITESPACES):
143 3 + 0
144 Error: Input contains invalid characters. Only digits (0-9), letters (a-z, A-Z)
       , and operators (+, -, *, /, %) are allowed.
145
146 Please enter (1) two integers or (2) one integer and one string (up to
147 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
148 line, separated by an operator (NO WHITESPACES):
149 abc+1
150 Operation type: +
151 Output String is: bcd
153 Please enter (1) two integers or (2) one integer and one string (up to
154 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
155 line, separated by an operator (NO WHITESPACES):
156 abc+26
157 Operation type: +
158 Output String is: abc
_{160} Please enter (1) two integers or (2) one integer and one string (up to
161 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
line, separated by an operator (NO WHITESPACES):
163 ABC+25
164 Operation type: +
165 Output String is: ZAB
167 Please enter (1) two integers or (2) one integer and one string (up to
168 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
169 line, separated by an operator (NO WHITESPACES):
170 abc-1
171 Operation type: -
172 Output String is: zab
174 Please enter (1) two integers or (2) one integer and one string (up to
175 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
```

```
176 line, separated by an operator (NO WHITESPACES):
177 abcd*2
178 Operation type: *
179 Output String is: abcdabcd
181 Please enter (1) two integers or (2) one integer and one string (up to
182 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
183 line, separated by an operator (NO WHITESPACES):
184 homework/3
185 Operation type: /
186 Output String is: homew
188 Please enter (1) two integers or (2) one integer and one string (up to
_{189} 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
190 line, separated by an operator (NO WHITESPACES):
191 homework/8
192 Operation type: /
193 Output String is:
195 Please enter (1) two integers or (2) one integer and one string (up to
196 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
197 line, separated by an operator (NO WHITESPACES):
198 10+20
199 Operation type: +
200 Output Number is: 30
202 Please enter (1) two integers or (2) one integer and one string (up to
203 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
204 line, separated by an operator (NO WHITESPACES):
205 100-50
206 Operation type: -
207 Output Number is: 50
208
209 Please enter (1) two integers or (2) one integer and one string (up to
_{210} 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
211 line, separated by an operator (NO WHITESPACES):
212 5*6
213 Operation type: *
214 Output Number is: 30
216 Please enter (1) two integers or (2) one integer and one string (up to
217 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
218 line, separated by an operator (NO WHITESPACES):
219 20/4
220 Operation type: /
221 Output Number is: 5
223 Please enter (1) two integers or (2) one integer and one string (up to
224 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
225 line, separated by an operator (NO WHITESPACES):
226 2147483647+0
227 Operation type: +
228 Output Number is: 2147483647
230 Please enter (1) two integers or (2) one integer and one string (up to
231 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
232 line, separated by an operator (NO WHITESPACES):
233 2147483647+1
234 Operation type: +
235 Error adding numbers; integer overflow detected.
237 Please enter (1) two integers or (2) one integer and one string (up to
238 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
```

```
line, separated by an operator (NO WHITESPACES):
10/0
211 Operation type: /
212 Error dividing numbers; division by zero.
213
214 Please enter (1) two integers or (2) one integer and one string (up to
215 100 characters, contains letters a-z and/or A-Z only). Enter everything in one
216 line, separated by an operator (NO WHITESPACES):
217 exit
218 Breaking out of Program...
```

Listing 13: tests.txt