CSE 230 Problem Set 10

# Problem 26.2: Step 1

Complete step 1 (and the 4 sub-steps) of the TDD process for a method in a class which stores a position on a chess board:

A chess board consists of 64 locations: 8 rows and 8 columns. Every column has a letter (a-h) and every row has a number (1-8). The user can use upper-case or lower-case letters and can even get the order mixed up. Thus, “c2” means the same thing as “2C” which is position 10. This is for the Coordinate::set(const char \*input) method.



Complete step 1: the requirements.

|  |
| --- |
| Requirements |
| Set method taking in strings like “c2”, “2C” |
| Member variable represents positions with an integer from 0 to 63 |
| Get method that returns a string in form “D4” |
|  |
|  |

# Problem 26.2: Step 2-5 for Bottom Left Corner

Complete step 2-5 of the TDD process for the “a1” test case:

Step 2: Write the test.

class Coordinate {

public:

Coordinate(){};

void set(const char \*input) {

};

private:

int currentPos;

}

Coordinate newCoordinate = new Coordinate();

newCoordinate.set("a1");

assert(newCoordinate.currentPos == 0);

newCoordinate.set("A1");

assert(newCoordinate.currentPos==0);

newCoordinate.set("1a");

assert(newCoordinate.currentPos==0);

newCoordinate.set("1A");

assert(newCoordinate.currentPos==0);

Step 3: Run the test (show the output here):

**Assertion failed: (newCoordinate.currentPos == 0), function main, file main.cpp, line 28.**

Step 4: Write the code:

class Coordinate {

public:

Coordinate(){};

void set(const char \*input) {

int row = -1;

int col = -1;

try {

if (nullptr == input) {

throw string("\tERROR: Please provide a valid string\n");

}

for (const char\* p = input; \*p; p++)

{

if (isalpha(\*p))

{

if (col != -1)

{

throw string("\tERROR: More than one column specifier\n");

}

else if (isupper(\*p))

{

char letter = \*p;

letter = (char)tolower(letter);

if ('a' <= letter && letter <= 'h')

col = letter - 'a';

else

{

throw string("\tERROR: Columns must be between a and h\n");

}

//throw string("\tERROR: Columns must be lowercase\n");

}

else if ('a' <= \*p && \*p <= 'h')

col = \*p - 'a';

else

{

throw string("\tERROR: Columns must be between a and h\n");

}

}

else if (isdigit(\*p))

{

if (row != -1)

{

throw string("\tERROR: More than one row specifier\n");

}

else if ('1' <= \*p && \*p <= '8')

row = \*p - '1';

else

{

throw string("\tERROR: Rows must be between 1 and 8\n");

}

}

else

{

throw string("\tERROR: Unknown letter\n");

}

}

if (row == -1)

{

throw string("\tERROR: You must specify a row\n");

}

else if (col == -1)

{

throw string("\tERROR: You must specify a column\n");

}

}

catch (string e) {

std::cout << e;

}

currentPos = row \* 8 + col;

};

int currentPos;

};

Step 5: Refactor:

#include <string>

#include <map>

#include <iostream>

using namespace std;

class Coordinate {

public:

Coordinate(){};

int currentPos;

void set(const char \*input) {

int row = -1;

int col = -1;

try {

if (nullptr == input) {

throw string("\tERROR: Please provide a valid string\n");

}

for (const char\* p = input; \*p; p++)

{

if (isalpha(\*p))

{

if (col != -1)

{

throw string("\tERROR: More than one column specifier\n");

}

else

{

char letter = \*p;

letter = (char)tolower(letter);

if ('a' <= letter && letter <= 'h') {

col = letter - 'a';

}

else {

throw string("\tERROR:Columns must be between a and h\n");

}

}

}

else if (isdigit(\*p))

{

if (row != -1)

{

throw string("\tERROR: More than one row specifier\n");

}

else if ('1' <= \*p && \*p <= '8')

row = \*p - '1';

else

{

throw string("\tERROR: Rows must be between 1 and 8\n");

}

}

else

{

throw string("\tERROR: Unknown letter\n");

}

}

if (row == -1)

{

throw string("\tERROR: You must specify a row\n");

}

else if (col == -1)

{

throw string("\tERROR: You must specify a column\n");

}

}

catch (string e) {

std::cout << e;

}

currentPos = row \* 8 + col;

};

};

int main() {

Coordinate newCoordinate = Coordinate();

newCoordinate.set("a1");

assert(newCoordinate.currentPos == 0);

newCoordinate.set("A1");

assert(newCoordinate.currentPos==0);

newCoordinate.set("1a");

assert(newCoordinate.currentPos==0);

newCoordinate.set("1A");

assert(newCoordinate.currentPos==0);

std::cout <<"Tests passed";

}

# Problem 26.3: Step 2-5 for Bottom Middle

Complete step 2-5 of the TDD process for the “c1” test case:

Step 2: Write the test.

Coordinate newCoordinate = Coordinate();

newCoordinate.set("c1");

assert(newCoordinate.currentPos == 2);

newCoordinate.set("C1");

assert(newCoordinate.currentPos==2);

newCoordinate.set("1c");

assert(newCoordinate.currentPos==2);

newCoordinate.set("1C");

assert(newCoordinate.currentPos==2);

std::cout <<"Tests passed";

Step 3: Run the test (show the output here):

Tests Passed

Step 4: Write the code:

#include <string>

#include <map>

#include <iostream>

using namespace std;

class Coordinate {

public:

Coordinate(){};

int currentPos;

void set(const char \*input) {

int row = -1;

int col = -1;

try {

if (nullptr == input) {

throw string("\tERROR: Please provide a valid string\n");

}

for (const char\* p = input; \*p; p++)

{

if (isalpha(\*p))

{

if (col != -1)

{

throw string("\tERROR: More than one column specifier\n");

}

else

{

char letter = \*p;

letter = (char)tolower(letter);

if ('a' <= letter && letter <= 'h') {

col = letter - 'a';

}

else {

throw string("\tERROR:Columns must be between a and h\n");

}

}

}

else if (isdigit(\*p))

{

if (row != -1)

{

throw string("\tERROR: More than one row specifier\n");

}

else if ('1' <= \*p && \*p <= '8')

row = \*p - '1';

else

{

throw string("\tERROR: Rows must be between 1 and 8\n");

}

}

else

{

throw string("\tERROR: Unknown letter\n");

}

}

if (row == -1)

{

throw string("\tERROR: You must specify a row\n");

}

else if (col == -1)

{

throw string("\tERROR: You must specify a column\n");

}

}

catch (string e) {

std::cout << e;

}

currentPos = row \* 8 + col;

};

};

Step 5: Refactor:

#include <string>

#include <map>

#include <iostream>

using namespace std;

class Coordinate {

public:

Coordinate(){};

int currentPos;

void set(const char \*input) {

int row = -1;

int col = -1;

try {

if (nullptr == input) {

throw string("\tERROR: Please provide a valid string\n");

}

for (const char\* p = input; \*p; p++)

{

if (isalpha(\*p))

{

if (col != -1)

{

throw string("\tERROR: More than one column specifier\n");

}

else

{

char letter = \*p;

letter = (char)tolower(letter);

if ('a' <= letter && letter <= 'h') {

col = letter - 'a';

}

else {

throw string("\tERROR:Columns must be between a and h\n");

}

}

}

else if (isdigit(\*p))

{

if (row != -1)

{

throw string("\tERROR: More than one row specifier\n");

}

else if ('1' <= \*p && \*p <= '8')

row = \*p - '1';

else

{

throw string("\tERROR: Rows must be between 1 and 8\n");

}

}

else

{

throw string("\tERROR: Unknown letter\n");

}

}

if (row == -1)

{

throw string("\tERROR: You must specify a row\n");

}

else if (col == -1)

{

throw string("\tERROR: You must specify a column\n");

}

}

catch (string e) {

std::cout << e;

}

currentPos = row \* 8 + col;

};

};

# Problem 26.4: Step 2-5 The rest of the requirements

Show all your unit tests:

|  |  |  |
| --- | --- | --- |
| Name | Input | Output |
| NORMAL COORDINATE | A1 | 0 |
|  | A2 | 1 |
|  | A3 | 2 |
|  | A4 | 3 |
|  | A5 | 4 |
|  | A6 | 5 |
|  | A7 | 6 |
|  | A8 | 7 |
|  | B1 | 8 |
|  | B2 | 9 |
|  | B3 | 10 |
|  | B4 | 11 |
|  | B5 | 12 |
|  | B6 | 13 |
|  | B7 | 14 |
|  | B8 | 15 |
|  | C1 | 16 |
|  | C2 | 17 |
|  | C3 | 18 |
|  | C4 | 19 |
|  | C5 | 20 |
|  | C6 | 21 |
|  | C7 | 22 |
|  | C8 | 23 |
|  | D1 | 24 |
|  | D2 | 25 |
|  | D3 | 26 |
|  | D4 | 27 |
|  | D5 | 28 |
|  | D6 | 29 |
|  | D7 | 30 |
|  | D8 | 31 |
|  | E1 | 32 |
|  | E2 | 33 |
|  | E3 | 34 |
|  | E4 | 35 |
|  | E5 | 36 |
|  | E6 | 37 |
|  | E7 | 38 |
|  | E8 | 39 |
|  | F1 | 40 |
|  | F2 | 41 |
|  | F3 | 42 |
|  | F4 | 43 |
|  | F5 | 44 |
|  | F6 | 45 |
|  | F7 | 46 |
|  | F8 | 47 |
|  | G1 | 48 |
|  | G2 | 49 |
|  | G3 | 50 |
|  | G4 | 51 |
|  | G5 | 52 |
|  | G6 | 53 |
|  | G7 | 54 |
|  | G8 | 55 |
|  | H1 | 56 |
|  | H2 | 57 |
|  | H3 | 58 |
|  | H4 | 59 |
|  | H5 | 60 |
|  | H6 | 61 |
|  | H7 | 62 |
|  | H8 | 63 |
| too many rows | G21 | ERROR: More than one row specifier |
| too many columns | GH3 | ERROR: More than one column specifier |
| two rows | 83 | ERROR: More than one row specifier |
| two columns | FA | ERROR: More than one column specifier |
| only one column | F | ERROR: You must specify a row |
| only one row | 4 | ERROR: You must specify a column |
| column out of range | I3 | ERROR: Columns must be between a and h |

Show the completed class:

class TestCoordinate {

public:

void run() {

Coordinate tester = Coordinate();

for (char row = '1'; row < '9'; row++) {

for (char col = 'a'; col < 'i'; col++) {

int position = (row - '1') \* 8 + (col - 'a');

string lowerAlphaNum = "";

lowerAlphaNum += col;

lowerAlphaNum += row;

string lowerNumAlpha = "";

lowerNumAlpha += row;

lowerNumAlpha += col;

char upper = (char)toupper(col);

string upperAlphaNum = "";

upperAlphaNum += upper;

upperAlphaNum += row;

string upperNumAlpha = "";

upperNumAlpha += row;

upperNumAlpha += upper;

tester.set(lowerAlphaNum.c\_str());

assert(tester.currentPos == position);

tester.set(lowerNumAlpha.c\_str());

assert(tester.currentPos == position);

tester.set(upperAlphaNum.c\_str());

assert(tester.currentPos == position);

assert(tester.getPosition() == upperAlphaNum.c\_str());

tester.set(upperNumAlpha.c\_str());

assert(tester.currentPos == position);

}

}

tester.set("11");

tester.set("AA");

tester.set("A21");

tester.set("1BB");

tester.set("C");

tester.set("1");

tester.set("I3");

std::cout << "Passed all tests" << std::endl;

}

};

Step 4: Write the code:

class Coordinate {

friend class TestCoordinate;

public:

Coordinate(){};

int currentPos;

std::string getPosition() {

int rowNum = floor(currentPos / 8);

int columnNum = currentPos % rowNum;

char row = '\0';

switch (rowNum) {

case 0:

row = 'A';

case 1:

row = 'B';

case 2:

row = 'C';

case 3:

row = 'D';

case 4:

row = 'E';

case 5:

row = 'F';

case 6:

row = 'G';

case 7:

row = 'H';

}

std::string returnCoordinate = std::to\_string(row) + std::to\_string(columnNum);

return returnCoordinate;

}

void set(const char \*input) {

int row = -1;

int col = -1;

try {

if (nullptr == input) {

throw string("\tERROR: Please provide a valid string\n");

}

for (const char\* p = input; \*p; p++)

{

if (isalpha(\*p))

{

if (col != -1)

{

throw string("\tERROR: More than one column specifier\n");

}

else

{

char letter = \*p;

letter = (char)tolower(letter);

if ('a' <= letter && letter <= 'h') {

col = letter - 'a';

}

else {

throw string("\tERROR:Columns must be between a and h\n");

}

}

}

else if (isdigit(\*p))

{

if (row != -1)

{

throw string("\tERROR: More than one row specifier\n");

}

else if ('1' <= \*p && \*p <= '8')

row = \*p - '1';

else

{

throw string("\tERROR: Rows must be between 1 and 8\n");

}

}

else

{

throw string("\tERROR: Unknown letter\n");

}

}

if (row == -1)

{

throw string("\tERROR: You must specify a row\n");

}

else if (col == -1)

{

throw string("\tERROR: You must specify a column\n");

}

}

catch (string e) {

std::cout << e;

}

currentPos = row \* 8 + col;

};

};

Step 5: Refactor

class Coordinate {

friend class TestCoordinate;

public:

Coordinate(){};

int currentPos;

std::string getPosition() {

int rowNum = floor(currentPos / 8);

int columnNum = currentPos % rowNum;

char row = '\0';

switch (rowNum) {

case 0:

row = 'A';

case 1:

row = 'B';

case 2:

row = 'C';

case 3:

row = 'D';

case 4:

row = 'E';

case 5:

row = 'F';

case 6:

row = 'G';

case 7:

row = 'H';

}

std::string returnCoordinate = std::to\_string(row) + std::to\_string(columnNum);

return returnCoordinate;

}

void set(const char \*input) {

int row = -1;

int col = -1;

try {

if (nullptr == input) {

throw string("\tERROR: Please provide a valid string\n");

}

for (const char\* p = input; \*p; p++)

{

if (isalpha(\*p))

{

if (col != -1)

{

throw string("\tERROR: More than one column specifier\n");

}

else

{

char letter = \*p;

letter = (char)tolower(letter);

if ('a' <= letter && letter <= 'h') {

col = letter - 'a';

}

else {

throw string("\tERROR:Columns must be between a and h\n");

}

}

}

else if (isdigit(\*p))

{

if (row != -1)

{

throw string("\tERROR: More than one row specifier\n");

}

else if ('1' <= \*p && \*p <= '8')

row = \*p - '1';

else

{

throw string("\tERROR: Rows must be between 1 and 8\n");

}

}

else

{

throw string("\tERROR: Unknown letter\n");

}

}

if (row == -1)

{

throw string("\tERROR: You must specify a row\n");

}

else if (col == -1)

{

throw string("\tERROR: You must specify a column\n");

}

}

catch (string e) {

std::cout << e;

}

currentPos = row \* 8 + col;

};

};