Problem 6.25.2. Implement a **integer matrix class**, as shown in the figure 6.38. You are given the test program in **intmatrix1test.cpp**.

intmatrix1 (Phase1) 1. You cannot use constructors/destructors/copy constructor/equal operators 2. main program is given to you in intmatrix1test.cpp. You cannot change anything 3. Need to write intmatrix1.h and intmatrix1.cpp Matrix initialization c.print(" Matrix c"); intmatri×1 a ; creates an empty matrix ---- Matrix c a.init(); $\alpha = []$ **Empty Matrix** g.print("Matrix g"); intmatrix1 b ; creates a 3 \times 4 matrix b.init(3,4); --- Matrix g -initialized to zero 1 2 0000 0000 3 4 5 6 0000 intmatrix1 e ; creates a 3 x 10 matrix e.init(3,10,7); initialized to 7 a.fini(); //Free memory and clean 777777777 777777777 isEmpty() 777777777 Return true if matrix is empty intmatrix1 c ; creates an empty matrix else return false c.init(2,0,7); as number of cols = 0 isEqual() intmatrix1 d ; creates an empty matrix d.init(0,10,7); as number of rows = 0 If matrix a == matrix b return true else return false creates a 2 x 3 matrix intmatrix1 f 1 2 f.init("1 2|3 4|5 6") add routine 1. Adds 2 matrices a and b(if possible) 5 6 2. Returns the answer(empty matrix creates a 2 x 3 matrix if not possible) intmatrix1 g ");12 g.init(" 1 2 | 3 4 | 5 6 mult routine 1. multiply 2 matrices a and b(if possible) intmatrix1 h; 2. Returns the answer(empty matrix h.init(" 1 2 |3 4 |5 6 8 ") creates an empty matrix if not possible) Never crashes. If not possible creates empty matrix

Figure 6.38: Integer matrix class(Phase 1)

```
1 /*-----
2 Copyright (c) 2013 Author: Jagadeesh Vasudevamurthy
3 file: intmatrix1test.cpp
5 On linux:
6 g++ intmatrix1.cpp intmatrix1test.cpp
7 valgrind a.out
9
10
11 /*-----
12 This file test intmatrix1 object
15 /*-----
16 All includes here
18 #include "intmatrix1.h"
19
20 /*-----
21 test init and fini
23 void test_init_fini() {
   intmatrix1 a ;
24
25
    a.init();
    a.print("Matrix a");
26
27
28
    intmatrix1 b ;
29
    b.init(3,4);
30
    b.print("Matrix b");
31
32
    intmatrix1 c ;
    c.init(2,0,7);
33
34
    c.print("Matrix c");
35
36
    intmatrix1 d ;
37
    d.init(0,10,7);
38
    d.print("Matrix d");
39
40
    intmatrix1 e ;
    e.init(3,10,7);
41
    e.print("Matrix e");
42
43
44
45
    intmatrix1 f ;
    f.init("1 2|3 4|5 6");
46
47
    f.print("Matrix e");
48
    intmatrix1 g ;
g.init(" 1 2 | 3 4 | 5 6 ") ;
g.print("Matrix g") ;
49
50
51
    assert(f.isEqual(g));
52
53
    assert(g.isEqual(f));
54
55
    intmatrix1 h;
    h.init(" 1 2 |3 4 |5 6 8 ");
h.print("Matrix h");
56
57
58
    assert(h.isEmpty());
59
    assert(!f.isEqual(h));
60
    a.fini();
61
    b.fini();
63
    c.fini();
64
    d.fini();
    e.fini();
65
66
    f.fini();
```

```
c:\work\c++\course\objects\intmatrix1\intmatrix1test.cpp
```

```
2
```

```
67
     g.fini();
 68 }
 69
 70 /*----
 71 test add1
 73 void test_add1(const char* as, const char* bs, const char* anss) {
     intmatrix1 a ;
 75
     a.init(as);
     a.print("Matrix a");
 76
 77
     intmatrix1 b ;
     b.init(bs);
 78
 79
     b.print("Matrix b");
 80
     intmatrix1 s = a.add(b);
     s.print("matrix s");
 81
 82
     intmatrix1 ans;
 83
     ans.init(anss);
 84
     ans.print("matrix expected ans");
 85
     assert(s.isEqual(ans));
 86
     assert(ans.isEqual(s));
 87
     a.fini();
     b.fini();
 88
 89
     s.fini()
 90
     ans.fini();
 91 }
 92
 93 /*-----
 94 test add
 95 -----*,
 96 void test_add() {
     test_add1("7 9 11|13 15 17 "," 6 8 10| 12 14 16 ",
                                                 "13 17 21 | 25 29 33") ;
 97
     test_add1("1 2 3 4 5 6 ","1 2 ","");
 98
 99 }
100
101 /*-----
103 -----*/
104 void test_mult1(const char* as, const char* bs, const char* anss) {
105
    intmatrix1 a ;
106
     a.init(as);
     a.print("Matrix a");
107
108
     intmatrix1 b ;
109
     b.init(bs);
110
     b.print("Matrix b");
111
     intmatrix1 s = a.mult(b);
112
     s.print("matrix s");
     intmatrix1 ans;
113
     ans.init(anss);
114
115
     ans.print("matrix expected ans");
116
     assert(s.isEqual(ans));
117
     assert(ans.isEqual(s));
118
     a.fini();
     b.fini();
119
120
     s.fini();
121
     ans.fini();
     cout <<"----\n" ;
122
123 }
124
125 /*-----
126 test mult
128 void test_mult() {
     test_mult1("1 2 3"," 2 1 3 | 3 3 2 | 4 1 2 ","20 10 13") ;
129
     test_mult1("3 4 2","13 9 7 15|8 7 4 6| 6 4 0 3 ","83 63 37 75");
130
     test_mult1("3","5 2 11|9 4 14","15 6 33|27 12 |42");
131
     const char* a = "3 9 0 2 2 9 5 2 0 2 2 1 9 6 6 8 7 5 6 1 4 9 8 9 3 3 2 9 2 1 7 4 1 9 0 1 2 9 5 2 4 2 \checkmark
132
```

```
0 3 7 3 9 1 5 9 0 6 6 7 8 2 9 3 4 6 8 4 9 1";
133
     const char* b = "6 1 6 0 8 3 0 0 6 8 9 0 6 6 7 2 4 8 2 0 5 4 6 7 2 4 4 2 2 6 9 8 4 8 2 2 4 6 4 1 1 5 🕊
     5 6 4 7 5 5 7 4 6 5 0 6 5 3 2 3 7 0 1 3 8 5";
134
     const char* s = "132 170 200 87 128 186 175 106|122 186 166 86 92 182 195 123|197 235 267 104 179 ✔
     243 253 178 128 140 164 63 86 162 194 140 118 164 184 85 110 174 166 98 138 142 144 83 91 162
     137 82 | 187 222 244 106 160 244 232 141 | 201 212 210 97 171 230 204 142";
135
     test_mult1(a,b,s);
136
     test_mult1("7 3|2 5 | 6 8| 9 0","8 14 0 3 1|7 11 5 91 3|8 4 19 5 57","");
137 }
138
139 /*-----
140 test bed
141 -----
142 void testbed() {
143
    test_init_fini();
144
     test_add();
     test_mult();
145
146 }
147
148 /*-----
149 main
150 ----
151 int main() {
152 testbed();
153
    return 0 ;
154 }
155
156 //EOF
157
158
159
```