

7.7 A set class

Definition
 What is a set? Well, simply put, it's a **collection**.

Set of prime numbers: {2, 3, 5, 7, 11, 13, 17}
Positive multiples of 3 that are less than 10: {3, 6, 9}

iset64

1. Empty set a = { }
2. The set we are implementing will have numbers between 0 to 63 only
3. In sets it does not matter what order the elements are in
 Example: {1,2,3,4} is the same set as {3,1,4,2}
4. Number of elements in the above set = 4
5. In our set minimum number of element is 0 - Empty set
 maximum number of element is 64
 and the elements will be between 0 to 63

1

Adding an element to set

```
a = {1,2}
a += 5 = {1,2,5}
a += {10,63} = {0,1,2,5,63}
```

3

Union of sets (overload with +)

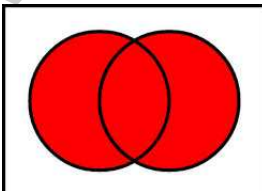
4
1 60

+

5
1

=

4 5
1 60



Intersection of sets (overload with *)

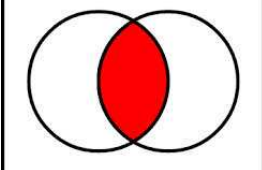
4
1 60

*

5
1

=

4
1 60



2

Removing an element

```
a = {1,6,10}
a -= 6 = {1,10}
```

5

Difference of sets (overload with -)

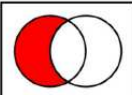
4
1 60

-

5
1

=

4
60



6

Is set equal? (overload ==)

1 4 6

==

4 1 6

What to submit?

1. iset64test.cpp cannot be modified. All tests must pass
2. Submit as a hardcopy
 1. iset64.h
 2. iset64.cpp
 3. Output as a pdf file
 4. A word doc that explains
 1. Data structure used
 2. Algorithms used for all the 6 methods above

Figure 7.14: Set

7.7. A SET CLASS

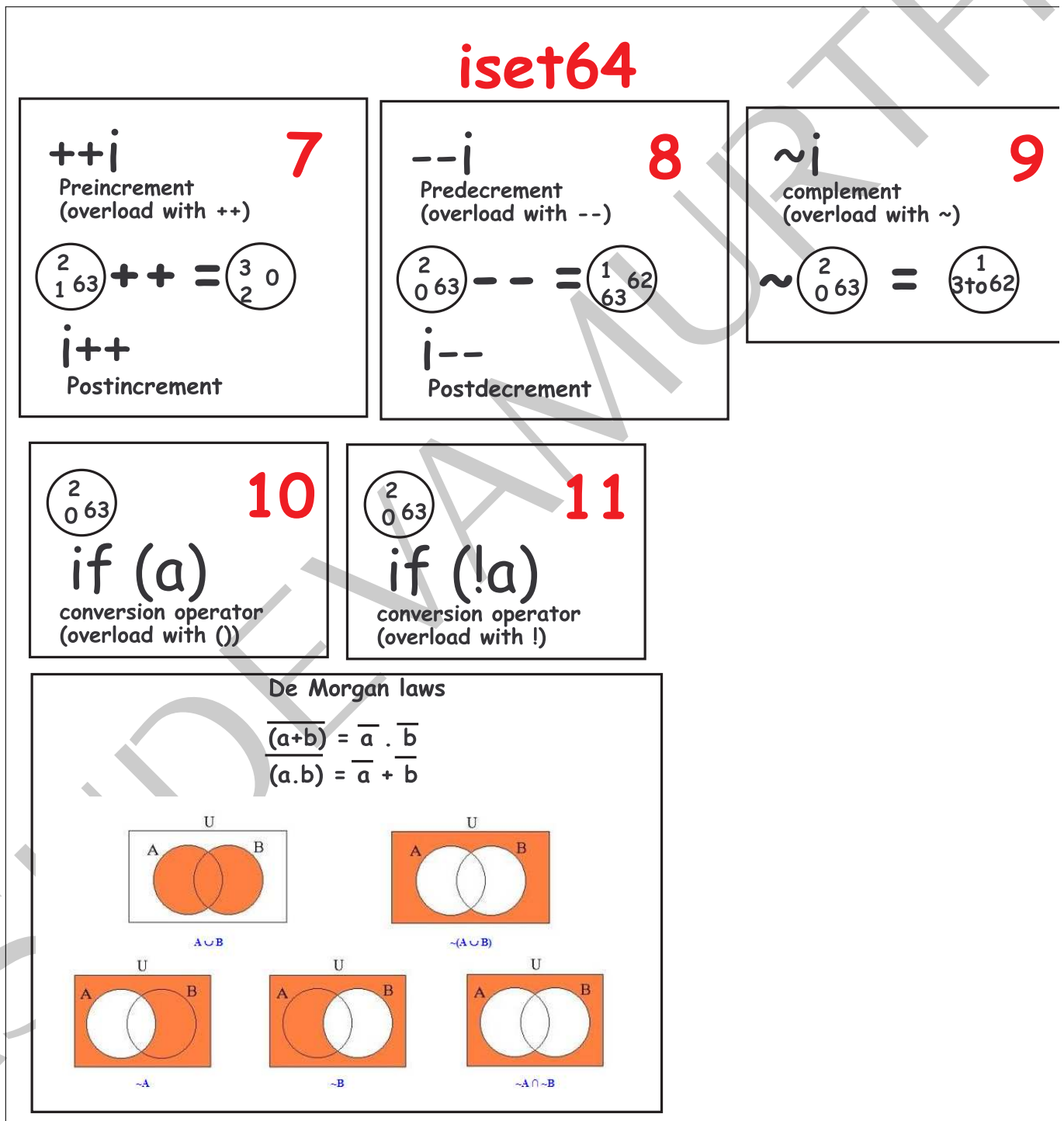


Figure 7.15: Set

```

1  /*-----
2  Copyright (c) 2013 Author: Jagadeesh Vasudevamurthy
3  file: iset64test.cpp
4
5  On linux:
6  g++ iset64.cpp iset64test.cpp
7  valgrind a.out
8
9  -----*/
10
11 /*-----
12 This file test iset64 object
13 -----*/
14
15 /*-----
16 All includes here
17 -----*/
18 #include "iset64.h"
19
20 /*-----
21 test a set
22 -----*/
23 void test_basic() {
24     iset64 a;
25     cout << "a = " << a << endl;
26     a = a + 5;
27     cout << "set a after adding 5 = " << a << endl;
28     a = a + 5;
29     cout << "set a after adding 5 = " << a << endl;
30     a += 63;
31     a += 0;
32     cout << "set a after adding 0 and 63 = " << a << endl;
33     int x[] = { 1, 3, 6 };
34     iset64 b(x, sizeof(x) / sizeof(int));
35     cout << "set b = " << b << endl;
36     b = b - 3;
37     cout << "set b after removing 3 = " << b << endl;
38     b = b - 3;
39     cout << "set b after removing 3 = " << b << endl;
40     b = b - 10;
41     cout << "set b after removing 10 = " << b << endl;
42     b = b - 6;
43     cout << "set b after removing 6 = " << b << endl;
44     b = b - 1;
45     cout << "set b after removing 1 = " << b << endl;
46     b = b + 10;
47     b = b + 2;
48     cout << "set b after adding {10,2} = " << b << endl;
49 }
50
51 /*-----
52 test union
53 -----*/
54 void test_union() {
55     {
56         cout << "TESTING: iset64 operator+(const iset64& a, const iset64& b)" << endl;
57         iset64 a;
58         a += 1;
59         a += 2;
60         iset64 b;
61         b += 1;
62         b += 2;
63         b += 3;
64         cout << "Set a " << a << endl;
65         cout << "Set b " << b << endl;
66         iset64 c = a + b;

```

```
67     cout << "a + b = " << c << endl;
68 }
69 {
70     cout << "TESTING:iset64 operator+(const iset64& a, const int b)" << endl;
71     iset64 a;
72     a += 1;
73     a += 2;
74     cout << a << endl;
75     a = a + 1;
76     cout << "{1,2} + 1 = " << a << endl;
77     a += 1;
78     a += 2;
79     cout << a << endl;
80     a = a + 3;
81     cout << "{1,2} + 3 = " << a << endl;
82 }
83 {
84     cout << "TESTING:iset64 operator+(const int b, const iset64& a)" << endl;
85     iset64 a;
86     a += 1;
87     a += 2;
88     cout << "Set a " << a << endl;
89     a = 1 + a;
90     cout << " 1 + {1,2} = " << a << endl;
91     a += 1;
92     a += 2;
93     cout << "Set a " << a << endl;
94     a = 3 + a;
95     cout << " 3 + {1,2} = " << a << endl;
96 }
97
98 {
99     cout << "TESTING:iset64& iset64::operator+=(const iset64& a)" << endl;
100    iset64 b;
101    b += 1;
102    b += 2;
103    iset64 a;
104    a += 1;
105    a += 3;
106    cout << "Set b " << b << endl;
107    cout << "Set a " << a << endl;
108    b += a;
109    cout << " {1,2} + {1,3} = " << b << endl;
110 }
111 {
112     cout << "iset64& iset64::operator+=(const int b)" << endl;
113     iset64 a;
114     a += 1;
115     a += 2;
116     cout << "Set a " << a << endl;
117     a += 3;
118     cout << " {1,2} + 3 = " << a << endl;
119 }
120 {
121     //test chaining
122     iset64 a;
123     a += 1;
124     a += 2;
125     iset64 b;
126     b += 3;
127     b += 4;
128     iset64 c;
129     c += 7;
130     c += 8;
131     iset64 d = a + b + c + 5;
132     cout << "Set a " << a << endl;
```

```
133     cout << "Set b " << b << endl;
134     cout << "Set c " << c << endl;
135     cout << "Set d " << d << endl;
136 }
137 }
138
139 /*-----
140 test difference
141 -----*/
142 void test_difference() {
143 {
144     cout << "TESTING: iset64 operator-(const iset64& a, const iset64& b)" << endl;
145     iset64 a;
146     a += 1;
147     a += 2;
148     iset64 b;
149     b += 1;
150     b += 2;
151     iset64 c = a - b;
152     cout << "Set a " << a << endl;
153     cout << "Set b " << b << endl;
154     cout << "a - b = " << c << endl;
155 }
156 {
157     cout << "TESTING: iset64 operator-(const iset64& a, const iset64& b)" << endl;
158     iset64 a;
159     a += 1;
160     a += 5;
161     iset64 b;
162     b += 1;
163     b += 2;
164     b += 3;
165     iset64 c = a - b;
166     cout << "Set a " << a << endl;
167     cout << "Set b " << b << endl;
168     cout << "a - b = " << c << endl;
169 }
170
171 {
172     cout << "TESTING: iset64 operator-(const iset64& a, const int b)" << endl;
173     iset64 a;
174     a += 1;
175     a += 2;
176     cout << "Set a " << a << endl;
177     a = a - 3;
178     cout << "a - 3 = " << a << endl;
179 }
180
181 {
182     cout << "TESTING: iset64 operator-(const int b, const iset64& a)" << endl;
183     iset64 a;
184     a += 1;
185     a += 2;
186     cout << "Set a " << a << endl;
187     a = 3 - a;
188     cout << "3 - a = " << a << endl;
189 }
190
191 {
192     cout << "TESTING: iset64& iset64::operator-=(const iset64& a)" << endl;
193     iset64 a;
194     a += 1;
195     a += 3;
196     iset64 b;
197     b += 1;
198     b += 2;
```

```

199     cout << "Set a " << a << endl;
200     cout << "Set b " << b << endl;
201     b -= a;
202     cout << "b -= a = " << b << endl;
203 }
204
205 {
206     cout << "TESTING: iset64& iset64::operator--(const int b)" << endl;
207     iset64 a;
208     a += 1;
209     a += 2;
210     cout << "Set a " << a << endl;
211     a -= 3;
212     cout << "a -= 3 = " << a << endl;
213 }
214 {
215     //test chaining
216     iset64 a;
217     a += 1;
218     a += 2;
219     iset64 b;
220     b += 2;
221     b += 4;
222     iset64 c;
223     c += 2;
224     c += 8;
225     iset64 d = a - b - c + 5;
226     cout << "Set a " << a << endl;
227     cout << "Set b " << b << endl;
228     cout << "Set c " << c << endl;
229     cout << "Set d " << d << endl;
230 }
231 }
232
233 /*-----
234 test intersection
235 -----*/
236 void test_intersection() {
237     {
238         cout << "TESTING: iset64 operator*(const iset64& a, const iset64& b)" << endl;
239         iset64 a;
240         a += 1;
241         a += 2;
242         iset64 b;
243         b += 1;
244         b += 2;
245         b += 3;
246         cout << "Set a " << a << endl;
247         cout << "Set b " << b << endl;
248         iset64 c = a * b;
249         cout << "a * b = " << c << endl;
250     }
251     {
252         cout << "TESTING: iset64 operator*(const iset64& a, const int b)" << endl;
253         iset64 a;
254         a += 1;
255         a += 2;
256         cout << "Set a " << a << endl;
257         a = a * 1;
258         cout << "{1,2} * 1 = " << a << endl;
259         a += 1;
260         a += 2;
261         cout << "Set a " << a << endl;
262         a = a * 3;
263         cout << "{1,2} * 3 = " << a << endl;
264     }

```

```

265 {
266     cout << "TESTING:iset64 operator*(const int b, const iset64& a)" << endl;
267     iset64 a;
268     a += 1;
269     a += 2;
270     cout << "Set a " << a << endl;
271     a = 1 * a;
272     cout << " 1 * {1,2} = " << a << endl;
273     a += 1;
274     a += 2;
275     cout << "Set a " << a << endl;
276     a = 3 * a;
277     cout << " 3 * {1,2} = " << a << endl;
278 }
279
280 {
281     cout << "TESTING:iset64& iset64::operator*=(const iset64& a)" << endl;
282     iset64 b;
283     b += 1;
284     b += 2;
285     iset64 a;
286     a += 1;
287     a += 3;
288     cout << "Set b " << b << endl;
289     cout << "Set a " << a << endl;
290     b *= a;
291     cout << " {1,2} * {1,3} = " << b << endl;
292 }
293 {
294     cout << "iset64& iset64::operator*=(const int b)" << endl;
295     iset64 a;
296     a += 1;
297     a += 2;
298     cout << "Set a " << a << endl;
299     a *= 3;
300     cout << " {1,2} * 3 = " << a << endl;
301 }
302 {
303     //test chaining
304     iset64 a;
305     a += 1;
306     a += 2;
307     iset64 b;
308     b += 2;
309     b += 4;
310     iset64 c;
311     c += 2;
312     c += 8;
313     iset64 d = a * b * c + 5;
314     cout << "Set a " << a << endl;
315     cout << "Set b " << b << endl;
316     cout << "Set c " << c << endl;
317     cout << "Set d " << d << endl;
318 }
319 }
320
321
322 /*-----
323 test equal
324 -----*/
325 void test_equal_not_equal() {
326     {
327         cout << "TESTING: bool operator==(const iset64& a, const iset64& b)" << endl;
328         iset64 a;
329         a += 1;
330         a += 2;

```



```

331     iset64 b;
332     b += 1;
333     b += 2;
334     cout << "Set a " << a << endl;
335     cout << "Set b " << b << endl;
336     cout << "a == b " << boolalpha << (a == b) << endl;
337     b -= 1;
338     cout << a;
339     cout << b;
340     cout << "a == b " << boolalpha << (a == b) << endl;
341 }
342 {
343     cout << "TESTING: bool operator!=(const iset64& a, const iset64& b)" << endl;
344     iset64 a;
345     a += 1;
346     a += 2;
347     iset64 b;
348     b += 1;
349     b += 2;
350     cout << "Set a " << a << endl;
351     cout << "Set b " << b << endl;
352     cout << "a != b " << boolalpha << (a != b) << endl;
353     b -= 1;
354     cout << "Set a " << a << endl;
355     cout << "Set b " << b << endl;
356     cout << "a != b " << boolalpha << (a != b) << endl;
357 }
358 }
359
360 /*-----
361 ++ and --
362 -----*/
363 void test_pre_post_inr_dec() {
364     {
365         int x[] = { 1, 2, 63 };
366         iset64 a(x, sizeof(x) / sizeof(int));
367         cout << "a = " << a << endl;
368         ++a;
369         cout << "++a = " << a << endl;
370         int y[] = { 2, 3, 0 };
371         iset64 b(y, sizeof(y) / sizeof(int));
372         assert(a == b);
373     }
374     {
375         int x[] = { 1, 2, 63 };
376         iset64 a(x, sizeof(x) / sizeof(int));
377         cout << "a = " << a << endl;
378         iset64 acopy(x, sizeof(x) / sizeof(int));
379         cout << "acopy = " << acopy << endl;
380         iset64 rhs = a++;
381         assert(rhs == acopy);
382         cout << "a++ = " << a << endl;
383         cout << "rhs = " << rhs << endl;
384         int y[] = { 2, 3, 0 };
385         iset64 b(y, sizeof(y) / sizeof(int));
386         assert(a == b);
387     }
388     {
389         int x[] = { 0, 2, 63 };
390         iset64 a(x, sizeof(x) / sizeof(int));
391         cout << "a = " << a << endl;
392         --a;
393         cout << "--a = " << a << endl;
394         int y[] = { 63, 1, 62 };
395         iset64 b(y, sizeof(y) / sizeof(int));
396         assert(a == b);

```

```

397     }
398     {
399         int x[] = { 0, 2, 63 };
400         iset64 a(x, sizeof(x) / sizeof(int));
401         cout << "a = " << a << endl;
402         iset64 acopy(x, sizeof(x) / sizeof(int));
403         cout << "acopy = " << acopy << endl;
404         iset64 rhs = a--;
405         assert(rhs == acopy);
406         cout << "a-- = " << a << endl;
407         cout << "rhs = " << rhs << endl;
408         int y[] = { 63, 1, 62 };
409         iset64 b(y, sizeof(y) / sizeof(int));
410         assert(a == b);
411     }
412 }
413
414 /*-----
415 ~
416 Complement of a set.
417 The complement of A is the set of all element in the universal set U, but not in A.
418 a = {0,2,63}
419 x = ~a
420 {1,3,...,62}
421 -----*/
422 void test_complement() {
423     {
424         int x[] = { 0, 2, 63 };
425         iset64 a(x, sizeof(x) / sizeof(int));
426         cout << "a = " << a << endl;
427         iset64 nota = (~a);
428         cout << "~a = " << nota << endl;
429         iset64 ans;
430         ans += 1;
431         for (int i = 3; i < 63; ++i) {
432             ans += i;
433         }
434         cout << "ans = " << ans << endl;
435         assert(nota == ans);
436         ans = ~ans;
437         cout << "~ans = " << ans << endl;
438         assert(ans == a);
439     }
440 }
441
442 /*-----
443 a = {0,2,63}
444 if (a) {
445
446 }
447 -----*/
448 void test_conversion_operator() {
449     int x[] = { 0, 2, 63 };
450     iset64 a(x, sizeof(x) / sizeof(int));
451     cout << "a = " << a << endl;
452     if (a) {
453         cout << "a exists\n";
454     } else {
455         cout << "a does not exists\n";
456     }
457     iset64 b;
458     cout << "b = " << b << endl;
459     if (b) {
460         cout << "b exists\n";
461     } else {
462         cout << "b does not exists\n";

```

```

463     }
464 }
465
466 /*-----
467 a = {0,2,63}
468 if (!a) {
469
470 }
471 -----*/
472 void test_not_operator() {
473     int x[] = { 0, 2, 63 };
474     iset64 a(x, sizeof(x) / sizeof(int));
475     cout << "a = " << a << endl;
476     if (!a) {
477         cout << "a does not exists\n";
478     } else {
479         cout << "a exists\n";
480     }
481     iset64 b;
482     cout << "b = " << b << endl;
483     if (!b) {
484         cout << "b does not exists\n";
485     } else {
486         cout << "b exists\n";
487     }
488 }
489
490 /*-----
491 (a+b)' = a'. b'
492 (a.b)' = a' + b'
493 -----*/
494 void test_demorgan_laws(const int x[], int lx, const int y[], int ly) {
495     {
496         iset64 a(x, lx);
497         cout << "a = " << a << endl;
498
499         iset64 b(y, ly);
500         cout << "b = " << b << endl;
501
502         iset64 aplusb = a + b;
503         cout << "apusb = " << aplusb << endl;
504
505         iset64 aplusbbar = ~(apusb);
506         cout << "apusbbar = " << aplusbbar << endl;
507
508         iset64 abar = ~(a);
509         cout << "abar = " << abar << endl;
510
511         iset64 bbar = ~(b);
512         cout << "bbar = " << bbar << endl;
513
514         iset64 abarplusbbar = abar + bbar;
515         cout << "abarplusbbar = " << abarplusbbar << endl;
516
517         iset64 abardotbbar = abar * bbar;
518         cout << "abardotbbar = " << abardotbbar << endl;
519
520         iset64 adotb = a * b;
521         cout << "adotb = " << adotb << endl;
522
523         iset64 adotbbar = ~(adotb);
524         cout << "adotbbar = " << adotbbar << endl;
525
526         assert(apusbbar == abardotbbar);
527         cout << "Demorgan law (a+b)' = a'. b' is proved\n";
528         assert(adotbbar == abarplusbbar);

```

```
529     cout << "Demorgan law (a.b)' = a' + b' is proved\n";
530 }
531 }
532
533 /*-----
534 (a+b)' = a'. b'
535 (a.b)' = a' + b'
536 -----*/
537 void test_demorgan_laws() {
538     {
539         int x[] = { 4, 5, 6 };
540         int y[] = { 5, 6, 8 };
541         test_demorgan_laws(x, (sizeof(x) / sizeof(int)), y, (sizeof(y) / sizeof(int)));
542     }
543     {
544         int x[] = { 1,2,4,5 };
545         int y[] = { 2,3,5,6 };
546         test_demorgan_laws(x, (sizeof(x) / sizeof(int)), y, (sizeof(y) / sizeof(int)));
547     }
548 }
549 }
550
551 /*-----
552 test bed
553 -----*/
554 void testbed() {
555     test_basic();
556     test_union();
557     test_difference();
558     test_intersection();
559     test_equal_not_equal();
560     test_pre_post_inr_dec();
561     test_complement();
562     test_conversion_operator();
563     test_not_operator();
564     test_demorgan_laws();
565 }
566
567 /*-----
568 main
569 -----*/
570 int main() {
571     testbed();
572     return 0;
573 }
574
575 //EOF
576
577
578
```