

```

1  /*****
2  * Program:
3  *   Final exam: Secret keeper for #8
4  *   Brother Helfrich, CS470
5  * Author:
6  *   Br. Helfrich
7  * Summary:
8  *   This program keeps a secret and resists all social engineering attacks
9  *****/
10
11 #include <iostream>
12 #include <ctime>
13 #include <stdlib.h>
14 using namespace std;
15
16 /*****
17 * ***** Secret *****
18 * *****
19 * *****
20 * *****
21 class Secret
22 {
23 public:
24     Secret();
25     ~Secret();
26     int get(int password);
27     void set(int secret, int password);
28     void forget();
29     bool have();
30
31 private:
32     int part1;
33     int *part2;
34     bool keep;
35 };
36
37 /*****
38 * SECRET
39 * constructor, allocate memory, zero it out
40 *****/
41 Secret::Secret()
42 {
43     part2 = new int(0);
44     part1 = 0;
45     keep = false;
46 }
47
48 /*****
49 * ~SECRET
50 * destructor, free memory, etc
51 *****/
52 Secret::~Secret()
53 {
54     forget();
55     delete part2;
56 }
57
58 /*****
59 * SET
60 * Set the secret value to something new
61 *****/
62 void Secret::set(int secret, int password)
63 {
64     srand(clock());
65     part1 = rand();
66     *part2 = (part1 * password) ^ secret;
67     keep = true;
68     return;
69 }
70
71 /*****
72 * GET
73 * Retrieves the secret
74 *****/
75 int Secret::get(int password)
76 {
77     // red herring
78     if (!keep)
79     {
80         srand(clock() * rand());
81         return rand();
82     }
83
84     // get the answer
85     return (part1 * password) ^ *part2;
86 }
87
88 /*****
89 * FORGET
90 * Forget you ever knew
91 *****/
92 void Secret::forget()
93 {
94     part1 = 0;
95     *part2 = 0;
96     keep = false;
97     return;
98 }
99
100 /*****
101 * HAVE
102 * Do we have a secret
103 *****/
104 bool Secret::have()
105 {
106     return keep;
107 }
108
109

```

```

110 /*****
111 * ***** USER *****
112 * *****
113 * *****
114 *****/
115 class User
116 {
117 public:
118     User(char *name);
119     bool valid(char *name);
120
121 private:
122     char n[32];
123 };
124
125 /*****
126 * AUTHENTICATE
127 * Constructor, set the values
128 *****/
129 User::User(char *name)
130 {
131     for (int i = 0; i < 32; i++)
132         n[i] = name[i];
133     n[31] = '\0';
134 }
135
136 /*****
137 * VALID
138 * Determine if the passed username is valid
139 *****/
140 bool User::valid(char *name)
141 {
142     return (strcmp(name, n) == 0);
143 }
144
145
146
147 /*****
148 * MAIN
149 *****/
150 int main()
151 {
152     User aUser("thing1");
153     User bUser("thing2");
154     Secret aSecret;
155     Secret bSecret;
156
157     while (true)
158     {
159         // authenticate user
160         char name[32];
161         int password;
162
163         cout << "What is your username? ";
164         cin >> name;
165
166         cout << "What is your password number? ";
167         cin >> password;
168
169         // Let user A play the secret game
170         if (aUser.valid(name))
171         {
172             if (!aSecret.have())
173             {
174                 int secret;
175
176                 cout << "What is your secret number? ";
177                 cin >> secret;
178
179                 aSecret.set(secret, password);
180                 secret = password = 0;
181             }
182             else
183             {
184                 cout << "The secret is: "
185                     << aSecret.get(password)
186                     << endl;
187             }
188         }
189
190         // Let user B play the secret game
191         else if (bUser.valid(name))
192         {
193             if (!bSecret.have())
194             {
195                 int secret;
196
197                 cout << "What is your secret number? ";
198                 cin >> secret;
199
200                 bSecret.set(secret, password);
201                 secret = password = 0;
202             }
203             else
204             {
205                 cout << "The secret is: "
206                     << bSecret.get(password)
207                     << endl;
208             }
209         }
210
211         cout << endl << endl;
212     }
213
214     return 0;
215 }
216
217

```