```cpp
#include <iostream>
using namespace std;

// Matthew Sanchez and Alexander Gomez
//
// CPSC 240
//

short a;
void base2()
{
    cout << "AX = ";
    short   x = 1 << 15, t, n = a;
    for (int i = 1; i <= 16; ++i)
    {
        t = n & x;
        if (t == 0)
        {
            cout << 0;
        }
        else
        {
            cout << 1;
        }
        if (i % 4 == 0)
        {
            cout << " ";
        }
        n = n << 1;
    }
    a = n; // save the original value of a
    cout << endl;
}


// Q1 functions and declarations

short numPrinters, numFloppyDrives, sizeOfRam, b;
void displayAll() {
    cout << "The number of printers connected to the computer: " << numPrinters
      << endl;
    cout << "The number if floppy drives: " << numFloppyDrives << endl;
    cout << "The size of RAM: " << sizeOfRam << endl;
}


// Q2 functions and declarations

short temp;
void beefIsValid() {
    cout << "BEEF is a valid ID for the family" << endl;
}
```

```
 52 void beefNotValid() {
 53     cout << "BEEF is not a valid ID for the family" << endl;
 54 }
 55 void fadeIsValid() {
 56     cout << "FADE is a valid ID for the family" << endl;
 57 }
 58 void fadeNotValid() {
 59     cout << "FADE is not a valid ID for the family" << endl;
 60 }
 61 void cabeIsValid() {
 62     cout << "CABE is a valid ID for the family" << endl;
 63 }
 64 void cabeNotValid() {
 65     cout << "CABE is not a valid ID for the family" << endl;
 66 }
 67
 68 // Q3 functions and declarations
 69
 70 short sprinklerCounter = 0, moveCounter = 0, maxMoves = 16, currentSprinkler =
      17;
 71 void displayNumSprinklers() {
 72     cout << sprinklerCounter << " sprinklers are ON" << endl;
 73 }
 74 void defectiveSprinklersSetup() {
 75     cout << "Defective sprinklers: ";
 76 }
 77 void displayCurrentSprinkler() {
 78     cout << currentSprinkler << " ";
 79 }
 80
 81 // Q4 functions and declarations
 82
 83 short currentFloor = 17;
 84 void elevatorSetup() {
 85     cout << "Elevator will stop at floors no. ";
 86 }
 87 void displayCurrentFloor() {
 88     cout << currentFloor << " ";
 89 }
 90
 91
 92 int main() {
 93
 94     // Q1
 95
 96     _asm {
 97         mov ax, 1100111010011100b;
 98         mov b, 0000000000001100;        // put in this binary to check bits 3 & 4
 99         and b, ax;                      // compares bits 3 and 4
100         shr b, 2;                       // moves out come to the right to compare
101         cmp b, 0;                       // the next few lines just compare all
              possibilities
```

```
102         Je ram16;                       // of the size of ram
103         cmp b, 1;
104         Je ram32;
105         cmp b, 2;
106         Je ram48;
107         cmp b, 3;
108         Je ram64;
109     ram16:
110         mov sizeOfRam, 16;
111         Jmp step2;
112     ram32:
113         mov sizeOfRam, 32;
114         Jmp step2;
115     ram48:
116         mov sizeOfRam, 48;
117         Jmp step2;
118     ram64:
119         mov sizeOfRam, 64;
120         Jmp step2;
121     step2:
122         mov b, 0000000011000000b;       // here we compare for the floppy drives
123         and b, ax;
124         shr b, 6;
125         cmp b, 0;
126         Je oneDrive;
127         cmp b, 1;
128         Je twoDrives;
129         cmp b, 2;
130         Je threeDrives;
131         cmp b, 3;
132         Je fourDrives;
133     oneDrive:                           // check all dloppy drive possibilites
134         mov numFloppyDrives, 1;
135         Jmp step3;
136     twoDrives:
137         mov numFloppyDrives, 2;
138         Jmp step3;
139     threeDrives:
140         mov numFloppyDrives, 3;
141         Jmp step3;
142     fourDrives:
143         mov numFloppyDrives, 4;
144         Jmp step3;
145     step3:                              // check bits 15 and 16 for number of
          printers connected
146         mov b, 1100000000000000b;
147         and b, ax;
148         shr b, 14;
149         cmp b, 0;                       // check all printer possibilities
150         Je noPrinters;
151         cmp b, 1;
152         Je onePrinter;
```

```
153         cmp b, 2;
154         Je twoPrinters;
155         cmp b, 3;
156         Je threePrinters;
157     noPrinters:
158         mov numPrinters, 0;
159         Jmp displayQ1;
160     onePrinter:
161         mov numPrinters, 1;
162         Jmp displayQ1;
163     twoPrinters:
164         mov numPrinters, 2;
165         Jmp displayQ1;
166     threePrinters:
167         mov numPrinters, 3;
168         Jmp displayQ1;
169     displayQ1:
170         call displayAll;
171     }
172
173     cout << endl << endl;
174
175     // Q2
176     _asm {
177         mov ax, 0xBEEF;                     // starting with BEEF base 16
178         mov temp, 0000000000000001b;
179         and ax, temp;
180         cmp ax, 0;                          // the only way a number can be odd ↵
              in binary
181         Je validBeef;                       // is if the digit on the far right ↵
              is 1
182         cmp ax, 1;                          // so we're checking for that for all ↵
               3 of our PINs
183         Je notValidBeef;
184     validBeef:
185         call beefIsValid;
186         Jmp checkFade;
187     notValidBeef:
188         call beefNotValid;
189     checkFade:
190         mov ax, 0;
191         mov ax, 0xFADE;
192         mov temp, 0000000000000001b;
193         and ax, temp;
194         cmp ax, 1;
195         Je notValidFade;
196         Jmp validFade;
197     notValidFade:
198         call fadeNotValid;
199         Jmp checkCabe;
200     validFade:
201         call fadeIsValid;
```

```
202     checkCabe:
203         mov ax, 0;
204         mov ax, 0xCABE;
205         mov temp, 0000000000000001b;
206         and ax, temp;
207         cmp ax, 1;
208         Je notValidCabe;
209         Jmp validCabe;
210     notValidCabe:
211         call cabeNotValid;
212         Jmp q2done;
213     validCabe:
214         call cabeIsValid;
215     q2done:
216     }
217
218     cout << endl << endl;
219
220     // Q3
221     _asm {
222         mov ax, 0110101000101111b;
223
224         // display as binary
225         mov a, ax;
226         call base2;
227         mov a, 0000000000000001b;
228
229         // check if sprinklers are on
230         mov bx, maxMoves;
231     startLoopQ3:
232         inc moveCounter;
233         cmp bx, moveCounter;
234         Je exitLoopQ3;
235         mov ax, 0110101000101111b;
236         and ax, a;
237         cmp ax, 0;
238         Jne increaseCounter;                // jump to increment sprinklerCounter ↪
                by 1
239         shl a, 1;                           // if not just shift left and go      ↪
              again
240         Jmp startLoopQ3;
241     increaseCounter:
242         inc sprinklerCounter;
243         shl a, 1;
244         Jmp startLoopQ3;
245     exitLoopQ3:
246         call displayNumSprinklers;
247     // find defective sprinklers
248         call defectiveSprinklersSetup;
249         mov a, 1000000000000000b;
250     startSecondLoopQ3:                      // here we're starting a loop to      ↪
            check
```

```
251         mov bx, 0;                          // each sprinkler one by one and starting with
252         dec currentSprinkler;               // 16 so we can display it in descending order
253         cmp bx, currentSprinkler;           // we initialize currentSprinkler to 17
254         Je exitSecondLoopQ3;                // so we can decrease and check at the beginning of the loop
255         mov ax, 0110101000101111b;
256         and ax, a;
257         cmp ax, 0;
258         Je isDefective;
259         shr a, 1;
260         Jmp startSecondLoopQ3;
261     isDefective:
262         call displayCurrentSprinkler;
263         shr a, 1;
264         Jmp startSecondLoopQ3;
265     exitSecondLoopQ3:
266     }
267
268     cout << endl << endl;
269
270     // Q4
271     _asm {
272         mov a, 1001000100001100b;
273         call base2;
274         call elevatorSetup;
275         mov a, 1000000000000000b;
276     startLoopQ4:                            // this is the same code as the last part
277         dec currentFloor;                   // of q3 except we have floors instead of
278         mov bx, 0;                          // sprinklers
279         cmp bx, currentFloor;
280         Je exitLoopQ4;
281         mov ax, 1001000100001100b;
282         and ax, a;
283         cmp ax, 0;
284         Jne willStop;
285         shr a, 1;
286         Jmp startLoopQ4;
287     willStop:
288         call displayCurrentFloor;
289         shr a, 1;
290         Jmp startLoopQ4;
291     exitLoopQ4:
292     }
293
294     cout << endl;
295
296
```

```
297     return 0;
298 }
```