（配点：(1) 20 点, (2-1) 20 点, (2-2) 20 点, (2-3) 20 点, (2-4) 20 点）

ハッシュ法（hash method）によって，いくつかの非負整数（nonnegative integer）を，添字（index）の範囲が 0〜$n-1$，要素数が n の配列（array）に格納（store）することを考える．ここで，格納する非負整数を n で割った剰余（remainder）を返すハッシュ関数（hash function）を用いて，ハッシュ値（hash value）を添字とするセル（cell）に非負整数を格納するものとする．なお，衝突（collision）が生じた際には，ハッシュ値に整数 k ($k \geq 1$) を加えて n で割った剰余を添字とするセルに格納する．それでもなお衝突が生じる場合には，非負整数が格納されていないセルが見つかるまで，k を $2k$, $3k$, $4k$, … と k の倍数へ順に置き換えて同様の処理を行うものとする．以下の各問に答えよ．

(1) $n = 10$, $k = 1$ とするハッシュ法を考える．どのセルにも非負整数が格納されていない状態から始めて，まず 21 を添字が 1 のセルに格納し，次に 15 を添字が 5 のセルに格納した状態を考える．これに続けて，28 と 35 をこの順番で格納するとき，それぞれどのセルに格納されるか，その添字を答えよ．

(2) 図 1 に示すC言語で書かれたプログラムは，あらかじめ 10 個の非負整数を配列に格納しておき，キーボードから入力された非負整数が配列に格納されているかどうかを，ハッシュ法によって調べるプログラムである．以下の各小問に答えよ．

(2-1) このプログラムが 43 行目まで実行された後の，`table[0]`〜`table[19]` の値を答えよ．なお，解答にあたっては，解答用紙の欄を用いること．

(2-2) 関数 `search(int *table, int d)` は，非負整数 `d` が配列 `table` に格納されていればその添字を，格納されていなければ `-1` を返す．そのような動作となるように，プログラム中の空欄（ア）〜（エ）を適切に埋めよ．

(2-3) このプログラムでは，`SKIP` は 3 となっている．これを 4 に変更すると，45 行目の `printf` 文以降が実行されない．このような問題が生じる理由を簡潔に答えよ．

(2-4) 小問 (2-3) のような問題が生じないようにするためには，一般に `MAX` と `SKIP` の間にどのような関係が成立していなければならないか，簡潔に答えよ．

```
1   #include <stdio.h>
2
3   #define EMPTY -1
4   #define MAX   20
5   #define SKIP   3
6
7   int hash(int x){
8     return x % MAX;
9   }
10
11  int next(int x){
12    x = x + SKIP;
13    return x % MAX;
14  }
15
16  void store(int *table, int d){
17    int h;
18    h = hash(d);
19    while (table[h] != EMPTY) h = next(h);
20    table[h] = d;
21  }
22
23  int search(int *table, int d){
24    int h;
25    h =  (ア) ;
26    while (table[h] != EMPTY) {
27      if (table[h] == d) return  (イ) ;
28      h =  (ウ) ;
29    }
30    return  (エ) ;
31  }
32
33  int main(){
34    int i, query;
35    int table[MAX];
36    int data[] = {31,45,59,25,95,39,76,27,65,43};
37
38    /* 配列を初期化する */
39    for (i=0; i<MAX; i++) table[i] = EMPTY;
40
41    /* 10 個の非負整数を格納する */
42    for (i=0; i<10;  i++) store(table, data[i]);
43
44    /* キーボードから非負整数を入力する */
45    printf("Query: ");
46    scanf("%d", &query);
47
48    /* 入力された非負整数が配列に格納されているかどうかを調べる */
49    if (search(table, query) != -1) printf("Found.\n");
50    else printf("Not found.\n");
51
52    return 0;
53  }
```

図 1: 非負整数が格納されているかどうかを調べるプログラム

(1) 28は添字0のセルに格納される.
35は添字6のセルに格納される.

(2) (2-1) table[0]~table[19] = {-1, -1, 39, 43, -1, 45, -1, 27, 25, -1, -1, 31, -1, -1, 65, 95, 76, -1, -1, 59}.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
-1	-1	39	43	-1	45	-1	27	25	-1	-1	31	-1	-1	65	95	76	-1	-1	59

(2-2) (ア) hash(h)
(イ) d
(ウ) next(h)
(エ) -1

(2-3) MAX=20, SKIP=4のとき、store(table, data[0]) ~ store(table, data[7])を実行して、
table[3]=39, table[7]=27, table[11]=31, table[15]=95, table[19]=59.
store(table, store[9])を実行するとき、hash(data[9])=hash(43)=3
このとき、19行目のwhileループ:
table[3]!=EMPTY、h=next(3)=7.
table[7]!=EMPTY、h=next(7)=11.
next(11)=15、next[15]=19、next(19)=3、next(3)=7…
このとき、whileループが無限ループとなる。45行目のprintf文以降が実行されない.

(2-4) MAXとSKIPが互いに素でなければならない.