ICP Midterm Solution

1 a) **`limits.h`** b) **`stdlib.h`** c) **`math.h`** d) **`stdio.h`**

2 a) ①
b) ①③

3 ① no
②③④ yes

4 a) **`size_t`**
b) **`unsigned`**

5 **`short x=(short)(sqrt((double)((int)x+2))+(double)x);`**

6 **`(n%2==0||n%3==0)&&n%6!=0`** // watch the parentheses

7 Because the evaluation order of the two operands of + is unspecified, the output may be
`SnoopyPluto11`
or
`PlutoSnoopy11`

8 Error: The function **`db`** is used before it is defined.
Correction: Either define **`db`** before **`main`** or declare the prototype of **`db`** before it is used in **`main`**. (N.B. The prototype may be declared either in the global scope or within the function **`main`**, as long as the declaration appears before **`db`** is used in **`main`**.)

9 a) **`i<n&&a[i]!=0`** // the order cannot be reversed
b) **`{ unsigned c=a%b; a=b; b=c; }`**
or
`{ unsigned c=a; a=b; b=c%b; }`
or
`{ unsigned c=b; b=a%b; a=c; }`
c) **`(b&1<<i)!=0`** or **`b&1<<i`** or **`(b>>i&1)==1`** or **`b>>i&1`**

10 a) `65535 ffffffff`

b) `1 4 7 3`

11 a) `177752`

b) `be4ccccd`

12 a)
```
***
**
*
*
```
b)
```
76
7654
765432
76543210
```

13 a) Loop A ① *k*+2 times ② *k*+1 times

Loop B ① 2 times ② *k*+1 times

b)
```
int a=0,b=0;
while (a++,n>0) {
    while (b++,n%2==0) n/=2;
    n--;
}
```
or
```
int a=1,b=0;
while (n>0) {
    a++; b++;
    while (n%2==0) { b++; n/=2; }
    n--;
}
```

14 a) Since the test `r*i<=UINT_MAX` never fails, the function won't return and the outer loop won't terminate.

b)
```
unsigned i;               // declare i outside the for loop
for (~~unsigned~~ i=2;i<=k;i++)
    if (r<=UINT_MAX/i) r*=i; else break;
if (i<=k) break;          // add this line
printf("%u!=%u\n",k,r);
```

15 a) It is inefficient in that each fib(k) is computed from scratch. It would be better if the value of fib(k) is obtained from the already-computed fib(k-1) and fib(k-2).

b)
```
int b(int n)
{
    ~~int k=1,a=fib(k);~~
    int k=1,a=1,b=1;
    while (n>=a) {
        n-=a; k++;
        ~~a=fib(k);~~ int c=a+b; a=b; b=c;
    }
    return k;
}
```

16 // Version A – Don't use return or break to exit a loop
```
bool distinct(int a[],int n)
{
    bool different=true;
    for (int i=0;i<n&&different;i++)
        for (int j=i+1;j<n&&different;j++)
            if (a[i]==a[j]) different=false;
    return different;
}
```
// Version B – Use return to exit a loop
```
bool distinct(int a[],int n)
{
    for (int i=0;i<n;i++)
        for (int j=i+1;j<n;j++)
            if (a[i]==a[j]) return false;
    return true;
}
```

(Cont'd on the next page)

16 (Cont'd)

// Version C – Use break to exit a loop

```
bool distinct(int a[],int n)
{
   int i;
   for (i=0;i<n;i++) {
      int j;
      for (j=i+1;j<n;j++)
         if (a[i]==a[j]) break;
      if (j<n) break;
   }
   return i==n;
}
```