UCLA Samueli School of Engineering

CS 33

MIDTERM EXAM

All answers should be written on the answer sheet.

All work should be written directly on the exam pages, use the backs of pages if needed.

This is an open book, open notes quiz – but you cannot share books or notes.

We will follow the departmental guidelines on reporting incidents of academic dishonesty.

Keep your eyes on your own exam!

NAME: _____

ID: _____

Problem 1: _____ (20)

Problem 2: _____ (25)

Problem 3: _____ (30)

Problem 4: _____ (25)

Total: _____ (out of 100)

1. ***This Problem Bytes (20 points)***: Consider two variables defined in a C program:

```
short x;
unsigned short ux;
```

The value of x is on your Answer Sheet. And suppose that ux is defined as:

```
ux=x;
```

This problem deals with what would be printed by the following statements:

```
a) printf("%u", ux);
b) printf("%d", (x>>1)^x);
c) printf("%d", ~x+1);
d) printf("%d", x<0U);
```

On the answer sheet, fill in each blank corresponding to the four statements above.

2. ***Pac Man Fever (40 points)***: Consider the following code:

```
int pacMan(int inky, int clyde) {

  int blinky, pinky;

  blinky = inky >> (clyde + (~0x00))
  pinky = !(~blinky);

  pinky = pinky | !blinky;

  return  pinky;
}
```

The variable and function names are not helpful in understanding what this code does. But your job is to figure out the intention of the code. For the two inputs, inky could be any integer and clyde will always be constrained to the following range:

```
1 <= clyde <= 32
```

Given the value of inky shown on the Answer Sheet, what values of clyde would result in function pacMan returning a 1? Express this as an inclusive range (e.g. 5-8 would mean that 5 <= clyde <=8 would result in a 1 being returned from pacMan).

3. ***Down in the Dumps (30 points)***: Suppose we have the following data structure declaration:

```
char * * grid[10][10];
```

This grid will point to different arrays of strings. The arrays of strings will be defined like this:

```
char * list0[10];
```

And then the grid will point to those arrays with assignments like this:

```
grid[3][4] = list0;
```

Your job is to answer the following questions (where the values of x, y, z are shown on the Answer Sheet):

a. What is the value of grid[x][y]?

b. What is the address pointed to by grid[x][y][z]?

c. What string would be printed by:

```
printf("%s", grid[x][y][z]);
```

The address of grid[0][0] is 0x601280 when compiled on an x86-64 architecture. To answer these questions, you may need the memory dumps below (these are taken from gdb, formatted exactly as we have been doing in class with the x command and the /xb formatting – the first column shows the starting address, and then the eight columns to the right of the address show the 8 bytes stored contiguously starting at that address):

Memory Dump #0:

```
0x601060 0xa0 0x08 0x40 0x00 0x00 0x00 0x00 0x00
0x601068 0xa4 0x08 0x40 0x00 0x00 0x00 0x00 0x00
0x601070 0xa9 0x08 0x40 0x00 0x00 0x00 0x00 0x00
0x601078 0xaf 0x08 0x40 0x00 0x00 0x00 0x00 0x00
0x601080 0xb6 0x08 0x40 0x00 0x00 0x00 0x00 0x00
0x601088 0xbc 0x08 0x40 0x00 0x00 0x00 0x00 0x00
0x601090 0xc2 0x08 0x40 0x00 0x00 0x00 0x00 0x00
0x601098 0xc8 0x08 0x40 0x00 0x00 0x00 0x00 0x00
0x6010a0 0xcd 0x08 0x40 0x00 0x00 0x00 0x00 0x00
0x6010a8 0xd4 0x08 0x40 0x00 0x00 0x00 0x00 0x00
0x6010b0 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x6010b8 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x6010c0 0xdb 0x08 0x40 0x00 0x00 0x00 0x00 0x00
0x6010c8 0xdf 0x08 0x40 0x00 0x00 0x00 0x00 0x00
0x6010d0 0xe6 0x08 0x40 0x00 0x00 0x00 0x00 0x00
```

```
0x6010d8 0xeb 0x08 0x40 0x00 0x00 0x00 0x00 0x00
0x6010e0 0xf0 0x08 0x40 0x00 0x00 0x00 0x00 0x00
0x6010e8 0xf6 0x08 0x40 0x00 0x00 0x00 0x00 0x00
0x6010f0 0xfc 0x08 0x40 0x00 0x00 0x00 0x00 0x00
0x6010f8 0x05 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x601100 0x0b 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x601108 0x10 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x601110 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x601118 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x601120 0x15 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x601128 0x1b 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x601130 0x21 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x601138 0x28 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x601140 0x2d 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x601148 0xcd 0x08 0x40 0x00 0x00 0x00 0x00 0x00
0x601150 0x33 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x601158 0x38 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x601160 0x3e 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x601168 0x46 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x601170 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x601178 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x601180 0x4b 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x601188 0x51 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x601190 0x56 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x601198 0x5b 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x6011a0 0x61 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x6011a8 0x66 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x6011b0 0x6a 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x6011b8 0x6f 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x6011c0 0x75 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x6011c8 0x7a 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x6011d0 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x6011d8 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x6011e0 0x82 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x6011e8 0x88 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x6011f0 0x8d 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x6011f8 0x93 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x601200 0x99 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x601208 0xa0 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x601210 0xa5 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x601218 0xaa 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x601220 0xaf 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x601228 0xb4 0x09 0x40 0x00 0x00 0x00 0x00 0x00
0x601230 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x601238 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x601240 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x601248 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
```

```
0x601250 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x601258 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x601260 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x601268 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x601270 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x601278 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x601280 0x80 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601288 0xc0 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601290 0x20 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601298 0x60 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x6012a0 0x80 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6012a8 0x60 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x6012b0 0xc0 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x6012b8 0x20 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6012c0 0xe0 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6012c8 0xc0 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x6012d0 0x20 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6012d8 0x20 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6012e0 0x60 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x6012e8 0xe0 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6012f0 0x80 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6012f8 0xc0 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601300 0x60 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601308 0xc0 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601310 0x20 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601318 0xc0 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601320 0xc0 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601328 0x80 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601330 0x20 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601338 0xe0 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601340 0x20 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601348 0x60 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601350 0x20 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601358 0x80 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601360 0x20 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601368 0x60 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601370 0xe0 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601378 0x20 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601380 0x20 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601388 0x80 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601390 0xe0 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601398 0x20 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6013a0 0x80 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6013a8 0xc0 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x6013b0 0xc0 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x6013b8 0x20 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6013c0 0xe0 0x11 0x60 0x00 0x00 0x00 0x00 0x00
```

```
0x6013c8 0x80 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6013d0 0xc0 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x6013d8 0xe0 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6013e0 0xe0 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6013e8 0x20 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6013f0 0x80 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6013f8 0xe0 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601400 0x60 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601408 0x60 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601410 0x80 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601418 0xc0 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601420 0xc0 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601428 0x60 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601430 0xc0 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601438 0x80 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601440 0x20 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601448 0x60 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601450 0xc0 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601458 0xc0 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601460 0x60 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601468 0x60 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601470 0xe0 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601478 0x20 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601480 0xc0 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601488 0x60 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601490 0xc0 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601498 0xe0 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6014a0 0x80 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6014a8 0x20 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6014b0 0xe0 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6014b8 0x60 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x6014c0 0x20 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6014c8 0x60 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x6014d0 0xe0 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6014d8 0x20 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6014e0 0xe0 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6014e8 0xe0 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6014f0 0x80 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x6014f8 0x60 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601500 0x20 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601508 0x80 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601510 0xc0 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601518 0x80 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601520 0x80 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601528 0xe0 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601530 0x80 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601538 0xc0 0x10 0x60 0x00 0x00 0x00 0x00 0x00
```

```
0x601540 0xe0 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601548 0xe0 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601550 0x20 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601558 0x60 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601560 0xc0 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601568 0x80 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601570 0xe0 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601578 0x20 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601580 0xc0 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601588 0xc0 0x10 0x60 0x00 0x00 0x00 0x00 0x00
0x601590 0xe0 0x11 0x60 0x00 0x00 0x00 0x00 0x00
0x601598 0xe0 0x11 0x60 0x00 0x00 0x00 0x00 0x00
```

Memory Dump #1:

```
0x4008a0: 0x72 0x65 0x64 0x00 0x62 0x6c 0x75 0x65
0x4008a8: 0x00 0x67 0x72 0x65 0x65 0x6e 0x00 0x79
0x4008b0: 0x65 0x6c 0x6c 0x6f 0x77 0x00 0x62 0x72
0x4008b8: 0x6f 0x77 0x6e 0x00 0x77 0x68 0x69 0x74
0x4008c0: 0x65 0x00 0x62 0x6c 0x61 0x63 0x6b 0x00
0x4008c8: 0x67 0x72 0x65 0x79 0x00 0x6f 0x72 0x61
0x4008d0: 0x6e 0x67 0x65 0x00 0x70 0x75 0x72 0x70
0x4008d8: 0x6c 0x65 0x00 0x74 0x65 0x61 0x00 0x63
0x4008e0: 0x6f 0x66 0x66 0x65 0x65 0x00 0x6d 0x69
0x4008e8: 0x6c 0x6b 0x00 0x73 0x6f 0x64 0x61 0x00
0x4008f0: 0x6a 0x75 0x69 0x63 0x65 0x00 0x77 0x61
0x4008f8: 0x74 0x65 0x72 0x00 0x73 0x6d 0x6f 0x6f
0x400900: 0x74 0x68 0x69 0x65 0x00 0x66 0x6c 0x6f
0x400908: 0x61 0x74 0x00 0x63 0x6f 0x6c 0x61 0x00
0x400910: 0x6d 0x61 0x6c 0x74 0x00 0x67 0x72 0x61
0x400918: 0x70 0x65 0x00 0x61 0x70 0x70 0x6c 0x65
0x400920: 0x00 0x62 0x61 0x6e 0x61 0x6e 0x61 0x00
0x400928: 0x70 0x65 0x61 0x72 0x00 0x6d 0x65 0x6c
0x400930: 0x6f 0x6e 0x00 0x70 0x6c 0x75 0x6d 0x00
0x400938: 0x70 0x65 0x61 0x63 0x68 0x00 0x61 0x70
0x400940: 0x72 0x69 0x63 0x6f 0x74 0x00 0x6b 0x69
0x400948: 0x77 0x69 0x00 0x68 0x6f 0x72 0x73 0x65
0x400950: 0x00 0x62 0x65 0x61 0x72 0x00 0x6c 0x69
0x400958: 0x6f 0x6e 0x00 0x74 0x69 0x67 0x65 0x72
0x400960: 0x00 0x77 0x6f 0x6c 0x66 0x00 0x63 0x6f
0x400968: 0x77 0x00 0x67 0x6f 0x61 0x74 0x00 0x73
0x400970: 0x68 0x65 0x65 0x70 0x00 0x64 0x65 0x65
0x400978: 0x72 0x00 0x67 0x6f 0x72 0x69 0x6c 0x6c
0x400980: 0x61 0x00 0x77 0x68 0x61 0x6c 0x65 0x00
0x400988: 0x73 0x65 0x61 0x6c 0x00 0x6f 0x74 0x74
```

```
0x400990: 0x65 0x72 0x00 0x73 0x68 0x61 0x72 0x6b
0x400998: 0x00 0x73 0x68 0x72 0x69 0x6d 0x70 0x00
0x4009a0: 0x63 0x72 0x61 0x62 0x00 0x63 0x6c 0x61
0x4009a8: 0x6d 0x00 0x66 0x69 0x73 0x68 0x00 0x66
0x4009b0: 0x72 0x6f 0x67 0x00 0x65 0x65 0x6c 0x00
0x4009b8: 0xbf 0x05 0x40 0x00 0x00 0x00 0x00 0x00
0x4009c0: 0xec 0x05 0x40 0x00 0x00 0x00 0x00 0x00
0x4009c8: 0x16 0x06 0x40 0x00 0x00 0x00 0x00 0x00
0x4009d0: 0x40 0x06 0x40 0x00 0x00 0x00 0x00 0x00
0x4009d8: 0x6a 0x06 0x40 0x00 0x00 0x00 0x00 0x00
0x4009e0: 0x25 0x64 0x20 0x25 0x64 0x20 0x25 0x64
0x4009e8: 0x20 0x25 0x6c 0x78 0x20 0x25 0x6c 0x78
0x4009f0: 0x20 0x3d 0x20 0x25 0x73 0x0a 0x00 0x00
0x4009f8: 0x01 0x1b 0x03 0x3b 0x44 0x00 0x00 0x00
0x400a00: 0x07 0x00 0x00 0x00 0x48 0xfa 0xff 0xff
0x400a08: 0x90 0x00 0x00 0x00 0x98 0xfa 0xff 0xff
0x400a10: 0x60 0x00 0x00 0x00 0x85 0xfb 0xff 0xff
0x400a18: 0xb8 0x00 0x00 0x00 0x9f 0xfc 0xff 0xff
0x400a20: 0xd8 0x00 0x00 0x00 0xe6 0xfc 0xff 0xff
0x400a28: 0xf8 0x00 0x00 0x00 0x18 0xfe 0xff 0xff
0x400a30: 0x18 0x01 0x00 0x00 0x88 0xfe 0xff 0xff
0x400a38: 0x60 0x01 0x00 0x00 0x00 0x00 0x00 0x00
0x400a40: 0x14 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x400a48: 0x01 0x7a 0x52 0x00 0x01 0x78 0x10 0x01
0x400a50: 0x1b 0x0c 0x07 0x08 0x90 0x01 0x07 0x10
0x400a58: 0x14 0x00 0x00 0x00 0x1c 0x00 0x00 0x00
0x400a60: 0x30 0xfa 0xff 0xff 0x2a 0x00 0x00 0x00
0x400a68: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x400a70: 0x14 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x400a78: 0x01 0x7a 0x52 0x00 0x01 0x78 0x10 0x01
0x400a80: 0x1b 0x0c 0x07 0x08 0x90 0x01 0x00 0x00
0x400a88: 0x24 0x00 0x00 0x00 0x1c 0x00 0x00 0x00
0x400a90: 0xb0 0xf9 0xff 0xff 0x50 0x00 0x00 0x00
0x400a98: 0x00 0x0e 0x10 0x46 0x0e 0x18 0x4a 0x0f
0x400aa0: 0x0b 0x77 0x08 0x80 0x00 0x3f 0x1a 0x3b
0x400aa8: 0x2a 0x33 0x24 0x22 0x00 0x00 0x00 0x00
0x400ab0: 0x1c 0x00 0x00 0x00 0x44 0x00 0x00 0x00
0x400ab8: 0xc5 0xfa 0xff 0xff 0x1a 0x01 0x00 0x00
0x400ac0: 0x00 0x41 0x0e 0x10 0x86 0x02 0x43 0x0d
0x400ac8: 0x06 0x03 0x15 0x01 0x0c 0x07 0x08 0x00
0x400ad0: 0x1c 0x00 0x00 0x00 0x64 0x00 0x00 0x00
0x400ad8: 0xbf 0xfb 0xff 0xff 0x47 0x00 0x00 0x00
0x400ae0: 0x00 0x41 0x0e 0x10 0x86 0x02 0x43 0x0d
0x400ae8: 0x06 0x02 0x42 0x0c 0x07 0x08 0x00 0x00
0x400af0: 0x1c 0x00 0x00 0x00 0x84 0x00 0x00 0x00
0x400af8: 0xe6 0xfb 0xff 0xff 0x2b 0x01 0x00 0x00
0x400b00: 0x00 0x41 0x0e 0x10 0x86 0x02 0x43 0x0d
```

```
0x400b08: 0x06 0x03 0x26 0x01 0x0c 0x07 0x08 0x00
0x400b10: 0x44 0x00 0x00 0x00 0xa4 0x00 0x00 0x00
0x400b18: 0xf8 0xfc 0xff 0xff 0x65 0x00 0x00 0x00
0x400b20: 0x00 0x42 0x0e 0x10 0x8f 0x02 0x45 0x0e
0x400b28: 0x18 0x8e 0x03 0x45 0x0e 0x20 0x8d 0x04
0x400b30: 0x45 0x0e 0x28 0x8c 0x05 0x48 0x0e 0x30
0x400b38: 0x86 0x06 0x48 0x0e 0x38 0x83 0x07 0x4d
0x400b40: 0x0e 0x40 0x6c 0x0e 0x38 0x41 0x0e 0x30
0x400b48: 0x41 0x0e 0x28 0x42 0x0e 0x20 0x42 0x0e
0x400b50: 0x18 0x42 0x0e 0x10 0x42 0x0e 0x08 0x00
0x400b58: 0x14 0x00 0x00 0x00 0xec 0x00 0x00 0x00
0x400b60: 0x20 0xfd 0xff 0xff 0x02 0x00 0x00 0x00
0x400b68: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
```

And here's an ASCII table if you need it:

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

4. ***Recursion Conundrum (25 points)***: Consider the following C code implementation of a Fibonacci term:

```
#include <stdlib.h>
#include <stdio.h>

unsigned int fib (unsigned int x)
{
  if (x==0)
    return 0;
  else if (x==1)
    return 1;
  else
    return fib(x-2)+fib(x-1);
}

int main(int argc, const char* argv[])
{
  int x;

  x=atoi(argv[1]);

  printf("%d %d\n", x, fib(x));

  return 1;
}
```

Here's the translated x86-64 assembly code for these functions:

```
000000000040057d <fib>:
  40057d:       55                      push   %rbp
  40057e:       53                      push   %rbx
  40057f:       48 83 ec 08             sub    $0x8,%rsp
  400583:       89 fb                   mov    %edi,%ebx
  400585:       b8 00 00 00 00          mov    $0x0,%eax
  40058a:       85 ff                   test   %edi,%edi
  40058c:       74 1b                   je     4005a9 <fib+0x2c>
  40058e:       b0 01                   mov    $0x1,%al
  400590:       83 ff 01                cmp    $0x1,%edi
  400593:       74 14                   je     4005a9 <fib+0x2c>
  400595:       8d 7f fe                lea    -0x2(%rdi),%edi
  400598:       e8 e0 ff ff ff          callq  40057d <fib>
  40059d:       89 c5                   mov    %eax,%ebp
  40059f:       8d 7b ff                lea    -0x1(%rbx),%edi
  4005a2:       e8 d6 ff ff ff          callq  40057d <fib>
  4005a7:       01 e8                   add    %ebp,%eax
  4005a9:       48 83 c4 08             add    $0x8,%rsp
  4005ad:       5b                      pop    %rbx
```

```
  4005ae:       5d                      pop    %rbp
  4005af:       c3                      retq

00000000004005b0 <main>:
  4005b0:       53                      push   %rbx
  4005b1:       48 8b 7e 08             mov    0x8(%rsi),%rdi
  4005b5:       ba 0a 00 00 00          mov    $0xa,%edx
  4005ba:       be 00 00 00 00          mov    $0x0,%esi
  4005bf:       e8 bc fe ff ff          callq  400480 <strtol@plt>
  4005c4:       48 89 c3                mov    %rax,%rbx
  4005c7:       89 c7                   mov    %eax,%edi
  4005c9:       e8 af ff ff ff          callq  40057d <fib>
  4005ce:       89 c2                   mov    %eax,%edx
  4005d0:       89 de                   mov    %ebx,%esi
  4005d2:       bf 80 06 40 00          mov    $0x400680,%edi
  4005d7:       b8 00 00 00 00          mov    $0x0,%eax
  4005dc:       e8 6f fe ff ff          callq  400450 <printf@plt>
  4005e1:       b8 01 00 00 00          mov    $0x1,%eax
  4005e6:       5b                      pop    %rbx
  4005e7:       c3                      retq
```

Suppose we set a breakpoint at 0x00000000004005a9 and then run the program. At the breakpoint, we examine memory at the current stack pointer (rsp). Using this information, can you determine what value of x will be printed in the statement from main():

```
printf("%d %d\n", x, fib(x));
```

This is the ***original*** value of x on the first invocation of fib(x) from main(). On the Answer Sheet, we have given you the result of the memory dump that you will need to find the value of x. Show your work to justify your answer.