MCS 494 midterm 1
2004.10.06
D. J. Bernstein

Your answers must be based solely on your own knowledge and the information on these sheets. You are not permitted to use books, notes, or computers. Do not ask your proctor for interpretations or clarifications.

Do not hand in this sheet. Anything that you want to have graded must appear in the answer booklet. Make sure that your name is on the front of the booklet.

Problem 1. The output of this program depends on the computer, the operating system, and the compiler:

```
int doit(int a,int b,int c,int d,int e,int f)
{
  int x[2];
  x[0] = 314;
  x[1] = 159;
  printf("%d %d %d %d %d %d %d %d\n"
    ,x[0],x[1],x[2],x[3],x[4],x[5],x[6],x[7]
    );
}
int main(void)
{
  doit(265,358,979,323,846,264);
}
```

Give an example of what the output might be; explain your answer. Give two more examples of what the output might be. Explain the differences between your three examples.

Problem 2. On my computer, this program runs forever:

```
int ptr;
one(int a)
{
  ptr = (&a)[-1];
}
two(int a)
{
  (&a)[-1] = ptr;
}
main()
{
  one(1);
  two(2);
}
```

Explain why, with pictures showing the contents of memory when `two` starts and when `two` returns.

Problem 3. A user sets up the following program to listen to network connections:

```
main()
{
  char line[512];
  gets(line);
  puts(line);
}
```

An attacker feeds this program the following input: 256 bytes of `90 90 90 90 90 90` etc. (byte `90` being a machine-language no-op); 256 bytes of machine-language instructions to make random changes to all of the user's files that haven't been read in the past month; and 256 bytes of `70 f8 bf bf 70 f8 bf bf 70 f8 bf bf` etc. Identify 10 possible memory locations for `line`, and 10 plausible memory locations for `main`'s return address, to make this attack work. Explain your answer.

Problem 4. Here are some instructions in x86 machine language, using registers `int cx` and `int *sp`:

```
D       sp += 0.25            move sp right 1 byte
I       --cx                  subtract 1 from cx
L       sp -= 0.25            move sp left 1 byte
Q       *--sp = cx            move sp left 4 bytes;
                              then copy cx to those 4 bytes
Y       cx = *sp++            copy cx from those 4 bytes;
                              then move sp right 4 bytes
hABCD   *--sp = 0x44434241    move sp left 4 bytes;
                              then copy 41 42 43 44 there
```

Give a sequence of these instructions to put bytes `40 3f 3e 3d 3c` into memory. Other effects are acceptable as long as that sequence of five bytes appears somewhere in memory.

Problem 5. The following function tries to guard against buffer overflows by checking for changes to a `canary` variable in the stack:

```
void printreverse()
{
  static int64 canaryok;
  int64 canary = random64();
  int xlen = 0;
  int x[32];
  canaryok = canary;
  while (scanf("%d",&x[xlen]) == 1) ++xlen;
  while (xlen > 0) printf("%d\n",x[--xlen]);
  if (canary != canaryok) abort();
}
```

Explain in detail how anyone who controls the input can cause the program to start following instructions at address `0x2809eedc` in memory. Assume that local variables are pushed onto the stack in the order that they are declared.