

Section 2 Extra Problems

Pointers

1. Determine the output of the following program.

```
#include <stdio.h>

int main(int argc, char **argv) {
    int *arr[4]; // assume arr stores the address 0x10
    int x = 4;   // assume &x = 0x40
    int y = 8;   // assume &y = 0x44

    long addr = (long) &arr[2];
    arr[2] = &x;
    arr[3] = arr[2];

    (*arr[3])++;
    arr[3]++;

    printf("addr = %lx\n", addr);
    printf("*arr[2] = %d\n", *arr[2]);
    printf("*arr[3] = %d\n", *arr[3]);
}
```

2. Write a function `reverse` that takes a string starting at a given `char*` and reverses the characters in the string. Hint: use a temporary `char *` to help you traverse the string, and remember that strings end with the null character `'\0'`.

```
void reverse(char *s) {
```

```
}
```

Bitshifting

1. Determine the output of the following program.

```
#include <stdio.h>

int main(int argc, char **argv) {
    int x = -1;
    short y = -1;
    x = x ^ 0xFFFF;
    y = y ^ 0xFFFF;

    printf("x = %d\n", x);
    printf("y = %hd\n", y); // %hd specifies a short
    printf("(short)x = %hd\n", (short)x);
}
```

2. The following program includes a buggy method `toggle_nth_bit` that doesn't work as it is intended to. First determine what is actually printed by `main` vs. what is intended to be printed. Then try to figure out the bug, and write a correct (and hopefully simpler) version of the method.

```
#include <stdio.h>

// Toggles the nth bit of the given
// val. Returns the toggled val
// Can assume 0 <= n <= 31
int toggle_nth_bit(int val, int n) {
    int one_farthest_left = 0x80000000;
    int mask = one_farthest_left >> (31 - n);
    return val ^ mask;
}

int main(int argc, char **argv) {
    int x = -1;

    printf("before toggle: %x\n", x);
    printf("after toggle: %x\n", toggle_nth_bit(x, 0));
}
```