

## EECS 110 Final March 17, 2008

***Don't panic!** Read each question through. If any part confuses you, ask me privately. Watch your time. Don't spend forever on any one question. Write cleanly. If you need to make big changes, X out your answer, write "see back" and write your new version on the back, with the number of the question.*

**1. (5 pts)** Show the output of the following program fragment:

| Program   | Output |
|---|--------|
| <pre>char * s = "abc"; char * t = "de"; char d[20]; char e[20];  printf("1. %s\n", strcpy(d, t)); printf("2. %s\n", strcat(d, s)); strcpy(e, d); printf("3. %s\n", strcat(e, d)); printf("4. %s\n", strchr(e, 'c')); printf("5. %s\n",        strstr(strchr(e, 'c'),              "de"));</pre> |        |

**2. (5 pts)** Show the output of the following program fragment:

| Program   | Output |
|---|--------|
| <pre>int a[5] = { 9, 8, 7, 6, 5 };  for (int * p = a + 4; p &gt; a; p--) {     printf("%d ", *p);     (*p)++; } printf("\n"); for (int i = 0; i &lt; 5; ++i) {     printf("%d ", a[i]); }</pre> |        |

**3. (10 pts)** The code below is supposed to read the following file of house data and print the summary on the right. The italicized comments do not appear in the data file.

| house.txt input file                  | Intended output   |
|---------------------------------------|---|
| LR 15 20 <i>15x20 living room</i>     | <b>5 rooms:</b><br><b>2 bedrooms</b><br><b>1 baths</b><br><b>Total area: 1026</b> |
| BR1 12 15 <i>12x15 first bedroom</i>  |   |
| BR2 12 10 <i>12x10 second bedroom</i> |   |
| BR3 -1 -1 <i>no third bedroom</i>     |   |
| KCHN 18 22 <i>18x22 kitchen</i>       |   |
| BATH1 5 6 <i>5x6 bathroom</i>         |   |
| BATH2 -1 -1 <i>no second bathroom</i> |   |

The code has many mistakes, both syntactic and logical. Circle every mistake and correct it with as small a change as possible. Focus just on this code, not on header files, prototypes, and functions called but not yet defined. Don't fix bad style, just errors.

```

typedef RoomType enum { "Bath", "Bedroom", "Other" };
typedef Room struct { RoomType type; int width, length; };

int main()
{
    int wid, len, num_rooms = 0, max_rooms = 15;
    char room_str[10];
    FILE fp = fopen("house.txt", "r");
    Room *rooms = (Room *) malloc(max_rooms * sizeof(RoomType));

    while (num_rooms < max_rooms && get_data(fp, room_str, wid, len)) {
        if (wid == -1 || len == -1) break;
        save_data(rooms[num_rooms], room_str, wid, len);
        ++num_rooms;
    }

    fclose(fp);
    printf("%d rooms:\n %d bedrooms\n %d baths\n Total area:  %d\n",
           num_rooms,
           count_rooms(rooms, num_rooms, Bedroom), /* count bedrooms */
           count_rooms(rooms, num_rooms, Bath),    /* count bathrooms */
           calculate_area(rooms, num_rooms));
    free(Room);
    return 0;
}
/* read data into variables; return false if no data found */
int get_data(FILE fp, char * room_str, int width, int length)
{
    return fscanf(fp, " %s %d %d", room_str, width, length);
}
/* put data into room structure */
void save_data(Room room, char * room_str, int width, int length)
{
    room.width = width;
    room.length = length;
    room.type = get_room_type(room_str);
}

```

**4. (5 pts)** Using `RoomType` and `Room` from Question 3, define a function `count_rooms(Room rooms[], int num_rooms, RoomType type)` to return the number of rooms in `rooms` of the given type, e.g., `count_rooms(rooms, num_rooms, Bedroom)` returns 2.

**5. (5 pts)** Using `RoomType` from Question 3, define `get_room_type(char *room_str)` to return the enumerated room type `Bedroom` for any room string starting with `BR`, `Bath` for any string starting with `BATH`, and `Other` for anything else. Use C's string library functions to do the comparisons, not a loop.

**6. (10 pts)** Define `is_palindrome(char *str)` to return true if and only if `str` is a palindrome, i.e., the same backwards as forwards, **ignoring** non-letters and case differences. Some palindromes: "go dog" "Pull up!" "A man, a plan, a canal – Panama!"

**Use only character functions, pointers and pointer arithmetic. No array notation, no string functions.** Do not create any new strings or arrays.