EECS 110 Final
March 12, 2007

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

`char s1[20] = "xyz";`
`char *s2 = "abcde";`
`char s3[20];`
 	
`printf("len s1 %d; s2 %d\n",`
`  strlen(s1), strlen(s2));`
`strcpy(s3, s2);`
`printf("s2 %s; s3 %s\n", s2, s3);`
`strcpy(s3, s1);`
`printf("s1 %s; s3 %s\n", s1, s3);`
`strcat(s3, s2);`
`printf("s2 %s; s3 %s\n", s2, s3);`
`printf("len s3 %d\n", strlen(s3));`
	`len s1 3; s2 5`
	`s2 abcde; s3 abcde`
	`s1 xyz; s3 xyz`
	`s2 abcde; s3 xyzabcde`
	`len s3 8`

Comment [CKR1]: A few people wrote 20 here.

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

Program	Output

`int a[5] = { 3, 4, 6, 2, 1 };`
`int *p = a;`
`int *q = a + 2;`
`int *r = &a[1];`
 	
`printf("%d %d\n", a[2], *(a + 2));`
`printf("%d %d\n", *p, *(p + 1));`
`printf("%d %d\n", *q, *(q + 1));`
`printf("%d %d\n", *r, *(r + 1));`
	`6 6`
	`3 4`
	`6 2`
	`4 6`

3. (15 pts) The code below is supposed to read a file of data on boats and print what it read. The file gives the number of boats, then lines with the type, name and speed, e.g.,

`boats.txt` input file	Program output
4 2 Breezy 4.5 0 Windy 12.2 1 HardWork 1.1 2 Chicago 3.9	2 Breezy 4.500000 0 Windy 12.200000 1 HardWork 1.100000 2 Chicago 3.900000

The code has more than a dozen mistakes. Circle every error (but only errors) you can find. Write the correct code next to it, making as small a change as possible.

```
typedef enumerated { Motor, Pedal Power, Sail } BOAT_TYPE;
typedef structure {
    BOAT_TYPE type;
    char * name;
    double speed;
} BOAT;

int main() {
    int num_boats;
    FILE *fp = fopen("boats.txt", "r");
    fscanf(fp, "%d", num_boats);
    BOAT boats[num_boats] = malloc(num_boats);
    read_boats(fp, boats, 4);
    print_boats(boats, 4);
    return 0;
}

void read_boats(FILE *fp, BOAT boats[], int size) {
    for (int i = 0; i < size; ++i) {
        read_boat(fp, boats[i]);
    }
}

void read_boat(FILE *fp, BOAT boat)
{
    fscanf(fp, "%c %s %lf", boat.type, boat.name, boat.speed);
}

void print_boats(BOAT boats[], int size)
{
    for (int i = 0; i < 4; ++i)
    {
        BOAT boat = boats[i];
        printf("%c %s %f\n", boat.type, boat.name, boat.speed);
    }
}
```

Comment [CKR2]: should be enum

Comment [CKR3]: should be PedalPower (spaces not legal in identifiers)

Comment [CKR4]: should be struct

Comment [CKR5]: should be char[n] where n is 9 or greater, otherwise no space reserved

Comment [CKR6]: missing prototypes for read_boats, read_boat, print_boats

Comment [CKR7]: should be &num_boats

Comment [CKR8]: A test for fp == NULL would be good.

Comment [CKR9]: should be *boats

Comment [CKR10]: in older compilers, should be (BOAT *)

Comment [CKR11]: should be num_boats * sizeof(BOAT)

Comment [CKR12]: Should be num_boats

Comment [CKR13]: Should be num_boats

Comment [CKR14]: Should have free(boats); fclose(fp); Note: not free(num_boats).

Comment [CKR15]: should be &boat[i], or boat + i, otherwise a copy will be filled, not the array

Comment [CKR16]: should be *boat

Comment [CKR17]: Some people put in a while loop here but that's wrong, because there's already a for loop in read_boats().

Comment [CKR18]: should be %d

Comment [CKR19]: should be &boat->type

Comment [CKR20]: should be boat->name (no &)

Comment [CKR21]: should be &boat->speed

Comment [CKR22]: should be size

Comment [CKR23]: should be %d

4. (5 pts) Define a function `member(n, data, size)` of three arguments, that, when passed an integer `n`, an array of integers `data`, and the `size` of the array, returns true if and only if `data` contains `n`, e.g., if the array `nums` is `[ 2, 5, 1, 9 ]` then `member(1, nums, 4)` returns true and `member(3, nums, 4)` returns false.

```
int member(int n, int nums[], int len)
{
    while (len-- > 0 && n != nums[len]) {}
    return len >= 0;
}
```

5. (10 pts) Define a function `histogram(data, size, h)` to take an array `data` of integers 0 through 9, the number `size` of integers in `data`, and an array `h` of size 10. `histogram()` should store in `h[0]` how many 0's were in `data`, in `h[1]` how many 1's were in `data`, and so on. For example, if `data` = `[2, 5, 1, 9, 2, 2, 3, 9, 0, 2, 5, 1, 2]`, then `h` should end up containing `[1, 2, 5, 1, 0, 2, 0, 0, 0, 2]`, because `data` has 1 zero, 2 1's, 5 2's, and so on.

```
void histogram(int h[11], int data[], int len)
{
    for (int i = 0; i < len; ++i)
    {
        h[data[i]]++;
    }
}
```

6. (10 pts) Define `strcut(dest, src, start, end)` to take a pointer to a character buffer `dest`, a pointer to a string `src`, a character `start` and a character `end`. `strcut()` should copy into `dest` a string with all the characters following the first occurrence of `start` in `src` up to the first occurrence of `end`, if any. `strcut()` should return `dest`. If `start` does not appear in `src`, then `dest` should contain an empty string. If `start` appears but `end` does not, everything after `start` to the end of `src` should be copied to `dest`. `strcut()` would be useful for getting text between things like parentheses and string quotes.This test code

```
char dest[20];
printf("%\"%s\"\n", strcut(dest, "abc<def>gh", '<', '>'));
printf("\"%s\"\n", strcut(dest, "abc<>gh", '<', '>'));
printf("\"%s\"\n", strcut(dest, "abcdef>gh", '<', '>'));
printf("\"%s\"\n", strcut(dest, "abc<def", '<', '>'));
```

would print

```
"def"
""
""
"def"
```

Use only pointers and pointer arithmetic. No array notation. Make sure `dest` always ends up containing a valid C string.

```
char * strcut(char *dest, char *src, char start, char end)
{
    char *temp = dest;
    // skip characters until start or null character
    while (*src != '\0' && *src != start) src++;
    // if start found...
    if (*src == start)
    {
        // go to next char...
        ++src;
        // copy until end or null character
        while (*src != '\0' && *src != end)
        {
            *temp++ = *src++;
        }
    }
    // add final null character and return dest
    *temp = '\0';
    return dest;
}
```

Comment [CKR24]: Code using library functions like strchr() and strncpy() is also fine, but it's not any simpler. A loop that uses a flag to determine whether to copy or not is also fine, though most versions ran through all of src when they didn't need to.

Comment [CKR25]: A temp pointer is needed, but a buffer of size 20 is wrong.

Comment [CKR26]: could be done with strchr() and a test for a NULL result.

Comment [CKR27]: Notice that even if nothing is copied at all, the null character is stored, to make sure dest is an empty string.