CS 241: System Programming Spring 2010
Instant C review

Answer the following questions about the given code segments.

1.
int p1;
What does **&p1** mean?

2.
```
char**argv;
```
What type is **argv**?
What type is ***argv**?
What type is ****argv**?

3.
```
main(int argc,char**argv) {
```
what is ***argv**?
what is **argv[argc]**?

4. What are the differences between **x** and **y**?
```
char* f() {
   char *x;
   static char*y;
   return y;
}
```

Fix the following code (if necessary)

5.
```
if(strcmp("a","a"))
printf("same!");
```

6.
```
int i =4;
int *iptr;
iptr = &i;
*iptr = 5;//now i=5
```

7.
```
char *p;
p=(char*)malloc(99);
strcpy("Hello",p);
printf("%s World",p);
free(p);
```

8.
```
char msg[5];
strcpy (msg,"Hello");
```

C Crib Notes
&: 'address-of' (reference operator)
***:** 'contents-of' (dereference operator)
Automatic variables are temporary and stored in the stack
char* p; **p** is a pointer to a character.
***p =0;** contents-of **p** set to 0. (Kaboom!)
After declaring a pointer, initialize it to something before using it. *(Doh!)*

C Strings
C strings are terminated with a null byte
strcpy("hello", "world") will crash
strcmp(s1,s2) returns 0 if same
argv[0] is the program name
argv[argc] is a null pointer

Memory allocation
malloc(bytes) to reserve heap memory
later… **free(ptr)**

static char* ptr;
static variables are not stored in stack

SegFault
Uninitialized pointers
strcpy(dest,"hello");
C Strings need null byte at the end

Buffer overflow
Un-initialized memory
Too confident: not checking return values
Miss-use of static vs. stack variables.