CS 341 #3 | *A day at the C Side*

Puzzle 1: What is the value of `p[strlen(p)]` **, if** `p` **points to** `"Sys"` **?**

Hint: `strlen("") ==0` `strlen("@")==1`, `strlen(NULL)==#@?#WT?!`

Q1: How do I find out how to use
___< *useful function or system call here*>__?

```
$
```

Puzzle 2: How do I find out how to use <u>`stat`</u> in C?
What are the manual sections?

Section 2? 3? 7?

Q2: For the start of the program, `main(int argc, char** argv)`

Sketch the argv memory structure
What is special about `argv[0]`
What is special about `argv[argc]`

How do you print out all of the arguments of a program?

```
int main(int argc, char**argv)
  for(int i=0;i<argc;i++) printf("%s;", argv[i])
//or...?



```

Q3: How do I allocate and free heap memory in C?

- Allocate:
- Free:

Can I make a pointer *really free* by freeing it twice?

What do we call a pointer that has been free'd?

Best Practice: <u>Always</u> set free'd pointers to NULL.

```
// ... code ...
free(ptr);
ptr = 0;
```

Q4 Puzzle 3: Fix a custom string concat (append) function:

```
 1: void mystrcat(char *dest, const char *src) {
 2:
 3:
 4:   while (*src) {
 5:
 6:
 7:     dest = src;
 8:
 9:
10:     src++; dest++;
11:
12:
13:   }
14: }
```

Puzzle 4 - Walk Through

Type	Variable	Memory Addr.
`const char *`	`src`	`0x1000`
`char *`	`dest`	`0x2000`

Address	Memory Contents
1000	'!'
1001	'2'
1002	'B'
1003	'\0'
…	
2000	'2'
2001	'B'
2002	'|'
2003	'\0'
2004	
2005	
2006	
…	

- ⇨ **Line 3:** What does `(*src)` do?
- ⇨ **Line 4:** What does `(dest = src)` do?
- ⇨ **Line 3..9:** When does the loop exit?

Q5. Puzzle 5: Fix my custom string duplication function

```
char *mystrdup(const char *src) {


  char *p = sizeof(src);


  strcpy(src, p);


  return p;
}
```

Q6: What is the purpose of a file stream, just files?
A "file stream" (or "file descriptor" in system calls) is the base interface to EVERYTHING external to RAM. This includes:

-
-
-
- Standard Streams:
    - **stdin**:
    - **stdout**:
    - **stderr**:

Q7: Writing to file streams: `fprintf`
What if the output of the following code snippet?

```
fprintf(stderr, "CS 341: ");
fprintf(stdout, "System ");
fprintf(stderr, "Programming ");
fprintf(stdout, "\n");
```

⇨ Result:

Q8: What is asprintf()?

```
int printf(const char * format, ...)
int fprintf(FILE * stream, const char * format, ...)
int sprintf(char * str, const char * format, ...);
int asprintf(char **strp, const char *fmt, ...)
```

⇨ **`char **strp`:**

⇨ **`const char *fmt`:**

Q9. Puzzle 6: Pointer Arithmetic

```
// Count the number of elements in an int-array
// until a number > 100  appears in the array:
int count (int *start) {
  int *ptr = start;





  return ___________ / ___________;
}
```

Q10 Debug Less: Use `assert` e.g. `assert(ptr && counter > 5);`
C provides the library macro assert that be used to find bugs in debugging and completely disappear in production code! Two modes:

- Debug mode (-g flag to add useful debugging info for the debugger):
- Production mode (#NDEBUG):

Best Practice: Always assert pre-conditions and assumptions.

Puzzle 7: Putting it altogether

```

// Sum an array of positive numbers, storing
// the result in `result` (by ref)
// and use asprintf to return a text version of result
char* mysum(const int *ptr,  int *result) {



  while ( *ptr ) {


    sum += *(ptr++);


  }
  char *text = NULL;

  asprintf(
  return text;
}
```