CSE 29 Week 3 Monday

void concat(char a[], char b[], char result[])

---

string.h

int strlen(char* str)

void strcpy(char* dest, char* src) copies chars from src to dest

void strcat(char* dest, char* src) append src to the end of dest

char str[] vs char* str

In function arguments, these do the same thing

void concat(char a[], char b[], char result[])
printf("%p %p %p\n", a, b, result);
// 0xffff1...bd 0xffff....d0 0xffff....d7
// represent an address in memory for that char data

void concat(char* a, char* b, char* result)
// same meanig

char* : an address in memory where we can access char data

T* : an address in memory where we can access T-typed data

Pointers = address data

Types:
- char
- int
- uint64_t
- int8_t

```c
#include <string.h>
#include <stdio.h>

// Takes two double arrays a and b, and changes result to have the
// appended array of a and b stored in it

// ASSUMES that result has enough space
void append(double a[], double b[], double result[]) {

}

void print_list(double lst[], int howmany) {
  for(int i = 0; i < howmany; i += 1) {
    printf("%f ", lst[i]);
  }
  printf("\n");
}


int main() {
  double n1[] = { 4.0, 5.0, 6.0 };
  double n2[] = { 0.7, 0.3, 0.1, 0.8 };
  double result[7];

  append(n1, n2, result);

  print_list(n1, 3);
  print_list(n2, 4);
}
```

for(int i=0; i < strlen(a); i+=1)

✗ char* expected, got double*

nothing says arrays are null-terminated!

C strings are null terminated!

```
void append(double a[], int alen,
            double b[], int blen,
            double result[])
```

Assume result has alen + blen elements of space

append(n1, 3, n2, 4, result);

↑ size/count

Variable/Role	Address	Data 0/8 1/9 2/A 3/B 4/C 5/D 6/E 7/F
	0x...88	
	0x...90	
	0x...98	
	0x...A0	
	0x...A8	
	0x...B0	
	0x...B8	
	0x...C0	
	0x...C8	
	0x...D0	
	0x...D8	
	0x...E0	
	0x...E8	
	0x...F0	
	0x...F8	
	0x...00	
	0x...08	
	0x...10	
	0x...18	
	0x...20	
	0x...28	
	0x...30	
	0x...38	
	0x...40	
	0x...48	
	0x...50	
	0x...58	
	0x...60	
	0x...68	
	0x...70	
	0x...78	
	0x...80	

```c
#include <stdio.h>
#include <string.h>

void concat(char* a, char* b, char* result) {
  printf("a: %p, b: %p\nresult: %p\n", a, b, result);
  int alen = strlen(a), blen = strlen(b);
  for(int i = 0; i < alen; i += 1) {
    result[i] = a[i];
  }
  for(int i = 0; i < blen; i += 1) {
    result[alen + i] = b[i];
  }
  result[alen + blen] = 0;
}

int main(int argc, char** argv) {
  printf("Argv: %p\n", argv);
  printf("Arg index 0: %p: %s\n", argv[0], argv[0]);
  printf("Arg index 1: %p: %s\n", argv[1], argv[1]);
  printf("Arg index 2: %p: %s\n", argv[2], argv[2]);

  char result[strlen(argv[1]) + strlen(argv[2]) + 1];
  concat(argv[1], argv[2], result);

  printf("%s\n", result);
}
```

argv is an address storing char* data
(char*)*

command-line arguments

```
$ gcc args.c -o args
$ ./args "Hello " "CSE29"
Argv: 0x7ffcda761c38
Arg index 0: 0x7ffcda7626e8: ./args
Arg index 1: 0x7ffcda7626ef: Hello
Arg index 2: 0x7ffcda7626f6: CSE29
a: 0x7ffe600146ef, b: 0x7ffe600146f6
result: 0x7ffe600130d0
Hello CSE29
```

Variable/Role	Address	Data 0/8 1/9 2/A 3/B 4/C 5/D 6/E 7/F
	0x...00	
	0x...08	
	0x...10	
	0x...18	
	0x...20	
	0x...28	
	0x...30	
c	0x...38	0x7f...e8
	0x...40	0x7f...ef
	0x...48	0x7f...f6
	0x...50	
	0x...58	
	0x...60	
	0x...68	
	0x...70	
	0x...78	
	0x...80	
	0x...88	
	0x...90	
	0x...98	
	0x...A0	
	0x...A8	
	0x...B0	
	0x...B8	
	0x...C0	
	0x...C8	
	0x...D0	
	0x...D8	/ a r g s \0 H
	0x...E0	e l l o ␣ \0 C S
	0x...E8	E 2 9 \0
	0x...F0	
	0x...F8	