Exam: many of you got "Try Again"

2: Full Pass

1: Partial Pass

0: Try Again

This is OK! "High standards, multiple tries"

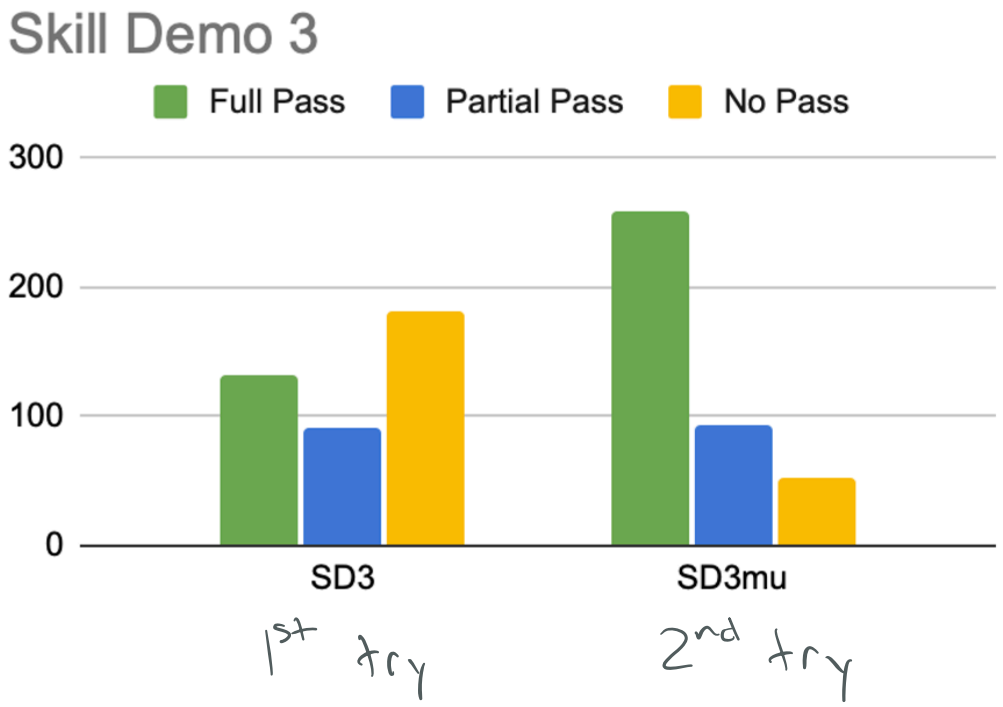
No permanent impact on your grade yet!

Exams 2+3 have same format - PrairieLearn

terminal environment

drawn from pset questions

CSE15L, 2023



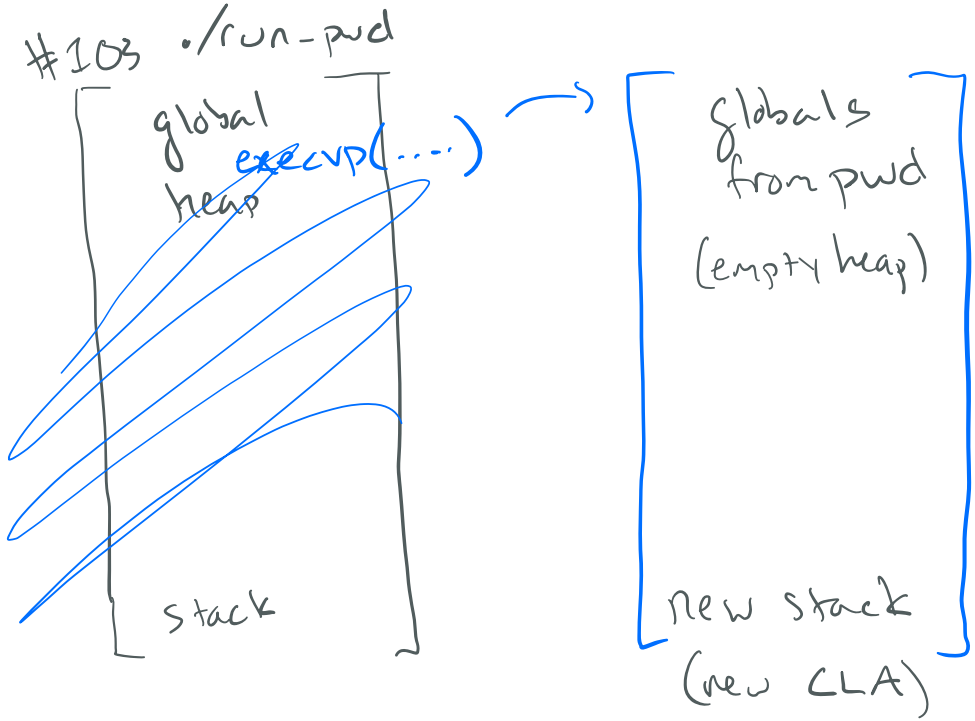
Lab is a great place for practice!

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

// execvp: system call that runs commands
// It *replaces* the current process by running the given command
//
// void execvp(char* command, char* args[])
// AND args is expected to be NULL terminated
// for example if we wanted to run `ls ..`
//
// execvp("ls", { "ls", "..", NULL })

// Write a program that calls the `pwd` (print working directory) command
int main() {
  char* args[] = { "pwd", NULL };
  execvp("pwd", args);
  printf("Hooray we did it!\n");
}
```

```
[jpolitz@ieng6-201]:ss1-25-05-w3t-malloc-heap:273$ pwd
/home/linux/ieng6/instructors/jpolitz/ss1-25-05-w3t-malloc-heap
[jpolitz@ieng6-201]:ss1-25-05-w3t-malloc-heap:274$ ./run_pwd ^C
[jpolitz@ieng6-201]:ss1-25-05-w3t-malloc-heap:274$ gcc run_pwd.c -o run_pwd
run_pwd.c: In function ‘main’:
run_pwd.c:15:3: warning: implicit declaration of function ‘printf’ [-Wimplicit
-function-declaration]
   15 |   printf("Hooray we did it!\n");
      |   ^~~~~~
run_pwd.c:2:1: note: include ‘<stdio.h>’ or provide a declaration of ‘printf’
    1 | #include <unistd.h>
  +++ |+#include <stdio.h>
    2 |
run_pwd.c:15:3: warning: incompatible implicit declaration of built-in functio
n ‘printf’ [-Wbuiltin-declaration-mismatch]
   15 |   printf("Hooray we did it!\n");
      |   ^~~~~~
run_pwd.c:15:3: note: include ‘<stdio.h>’ or provide a declaration of ‘printf’
[jpolitz@ieng6-201]:ss1-25-05-w3t-malloc-heap:275$ gcc run_pwd.c -o run_pwd
[jpolitz@ieng6-201]:ss1-25-05-w3t-malloc-heap:276$ ./run_pwd
/home/linux/ieng6/instructors/jpolitz/ss1-25-05-w3t-malloc-heap
[jpolitz@ieng6-201]:ss1-25-05-w3t-malloc-heap:277$
```

no printf output!

execvp: replaces the current process ("takes over") with the given command



execvp() vs. fork()

execvp:

Replaces current process with different code/behavior

fork:

Creates a new process with the same code/behavior

"forkexec": creates a new process with new code/behavior

- common pattern: call fork(), then in child process (where fork() returned 0), call exec

(Olivia has recording of this)

```c
// Write a function get_middle_i(int32_t* nums) that returns the middle
// number (by index) of an array of numbers that is expected to be nonempty
int32_t get_middle_i(int32_t* nums) {

}
```

There is no way in C to get the size of an array from its address.

NOT POSSIBLE TO WRITE!

Solution: pass size information along w/arrays.

```c
int32_t get_middle_i(int32_t* nums, uint32_t size) {
        return nums[size / 2];
}
```

%d : 32-bit #s
%ld : 64-bit #s

print addr as num

(Pointers)
Addresses are always 8 bytes long

```
int32_t* nums = NULL;
printf("%ld %p\n", (int64_t)nums, nums);
```

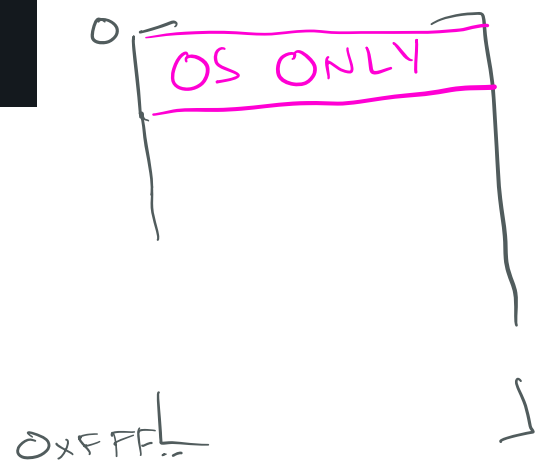
```
[jpolitz@ieng6-201]:ss1-25-05-w3t-malloc-heap:295$ ./get_middle
0 (nil)
```

```
// Write a function get_middle_i(int32_t* nums) that returns the middle
// number (by index) of an array of numbers that is expected to be nonempty
int32_t get_middle_i(int32_t* nums, uint32_t size) {
        return nums[size / 2];
}


int main() {

        int32_t* nums = NULL;
        printf("%ld %p\n", (int64_t)nums, nums);

        printf("%d\n", get_middle_i(nums, 0));
```

```
[jpolitz@ieng6-201]:ss1-25-05-w3t-malloc-heap:297$ ./get_middle
0 (nil)
Segmentation fault (core dumped)
```

nums was NULL (or 0)

Looking up address 0!



adds debug

```
$ gcc get_middle.c -g -o get_middle
```

```
$ valgrind ./get_middle
```

run w/valgrind

line # of "seg fault" invalid access

stack trace

```
==2901459== Invalid read of size 4
==2901459==    at 0x1091C8: get_middle_i (get_middle.c:15)
==2901459==    by 0x10920F: main (get_middle.c:24)
==2901459==  Address 0x0 is not stack'd, malloc'd or (recently) free'd
==2901459==
==2901459==
==2901459== Process terminating with default action of signal 11 (SIGSEGV)
==2901459==  Access not within mapped region at address 0x0
==2901459==    at 0x1091C8: get_middle_i (get_middle.c:15)
==2901459==    by 0x10920F: main (get_middle.c:24)
```

```
// A function, let's call it concat, that takes two strings
// and returns a new string
//
// Give me a function signature for concat(): return type, argument types,
// etc

// Options:

// This version “works”, is not in the spirit of the Java/Python examples
void concat_res(char str1[], char str2[], char result[])

// The array-style signature is not allowed
char[] concat_arr(char str1[], char str2[])

// Let's do this version
char* concat(char str1[], char str2[])
```

```c
char* concat(char str1[], char str2[]) {
        // THERE IS BAD STUFF IN HERE
        int l1 = strlen(str1);
        int l2 = strlen(str2);
        char result[l1 + l2 + 1];
        for(int i = 0; i < l1; i += 1) {
                result[i] = str1[i];
        }
        for(int j = l1; j < l1 + l2; j += 1) {
                result[j] = str2[j - l1];
        }
        result[l1 + l2] = '\0';
        char* toReturn = result;
        printf("The newly created string is at %p\n", toReturn);
        return toReturn;
}

int main() {
        printf("Should be abcdef: %s\n", concat("abc", "def"));
        printf("Should be hello world: %s\n", concat("hello", " world"));
        char* test3 = concat("a", "b");
        char* test4 = concat("zzz", "");
        char* test5 = concat("", "1234567");
        printf("Should be ab: %s\n", test3);
        printf("Should be zzz: %s\n", test4);
        printf("Should be 1234567: %s\n", test5);
}
```

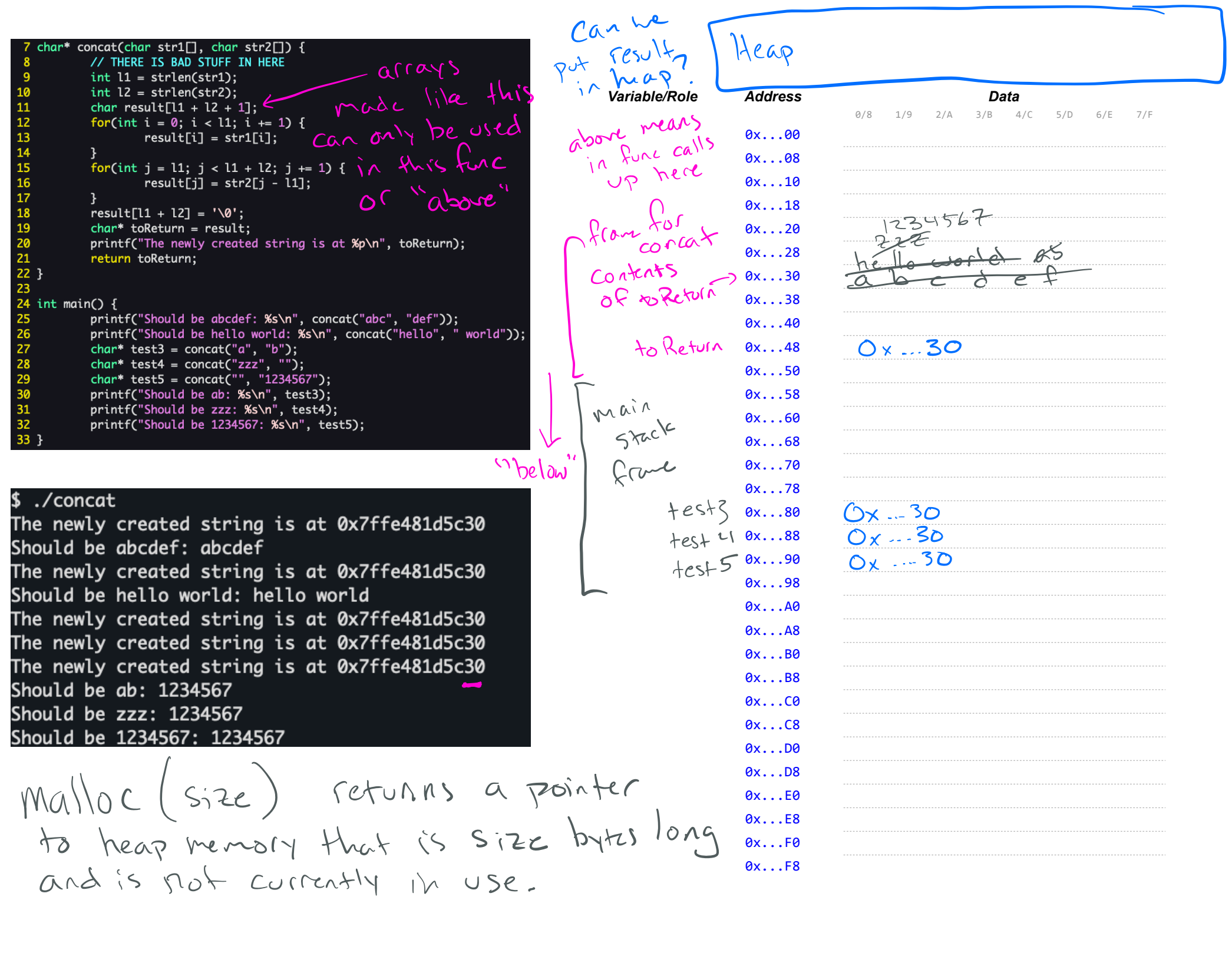
arrays made like this can only be used in this func or "above"

```
$ ./concat
The newly created string is at 0x7ffe481d5c30
Should be abcdef: abcdef
The newly created string is at 0x7ffe481d5c30
Should be hello world: hello world
The newly created string is at 0x7ffe481d5c30
The newly created string is at 0x7ffe481d5c30
The newly created string is at 0x7ffe481d5c30
Should be ab: 1234567
Should be zzz: 1234567
Should be 1234567: 1234567
```

malloc(size) returns a pointer to heap memory that is size bytes long and is not currently in use.

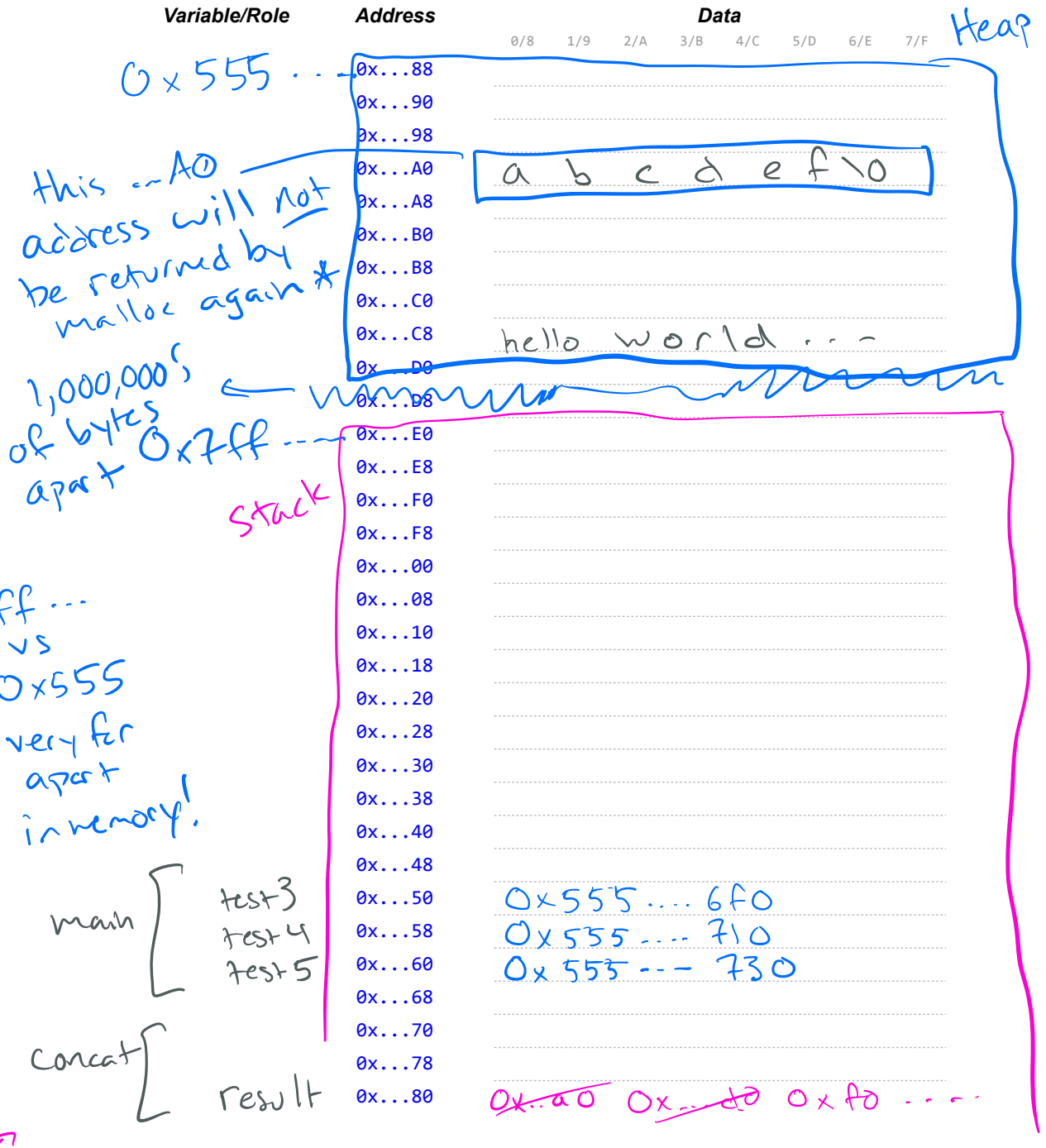
Can we put result in heap?

"below"



```c
// Let's do this version: fully in spirit of Java/Python
char* concat(char str1[], char str2[]) {
        // THERE IS BAD STUFF IN HERE
        int l1 = strlen(str1);
        int l2 = strlen(str2);
        char* result = malloc(l1 + l2 + 1);
        // char result[l1 + l2 + 1];  (BAD when returned)
        for(int i = 0; i < l1; i += 1) {
                result[i] = str1[i];
        }
        for(int j = l1; j < l1 + l2; j += 1) {
                result[j] = str2[j - l1];
        }
        result[l1 + l2] = '\0';
        printf("This stack frame is around %p\n", &result);
        printf("The newly created string is at %p\n", result);
        return result;
}

int main() {
        printf("Should be abcdef: %s\n", concat("abc", "def"));
        printf("Should be hello world: %s\n", concat("hello", " world"));
        char* test3 = concat("a", "b");
        char* test4 = concat("zzz", "");
        char* test5 = concat("", "1234567");
        printf("Should be ab: %s\n", test3);
        printf("Should be zzz: %s\n", test4);
        printf("Should be 1234567: %s\n", test5);
}
```

```
$ ./concat
This stack frame is around 0x7ffd36152e80
The newly created string is at 0x55584edd32a0
Should be abcdef: abcdef
This stack frame is around 0x7ffd36152e80
The newly created string is at 0x55584edd36d0
Should be hello world: hello world
This stack frame is around 0x7ffd36152e80
The newly created string is at 0x55584edd36f0
This stack frame is around 0x7ffd36152e80
The newly created string is at 0x55584edd3710
This stack frame is around 0x7ffd36152e80
The newly created string is at 0x55584edd3730
Should be ab: ab
Should be zzz: zzz
Should be 1234567: 1234567
```



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								
	0x...38								
	0x...40								
	0x...48								
	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								
	0x...E0								
	0x...E8								
	0x...F0								
	0x...F8								