CS 214 Recitation (Sec.7)

Oct. 3, 2017

Topic of this week

- Malloc's placement strategies
- Valgrind

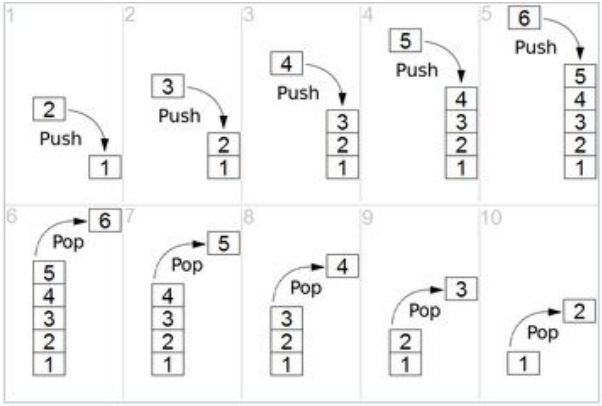
Memory allocation

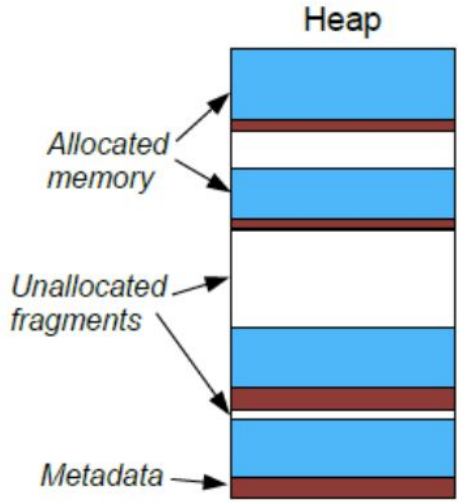
- (stack): stack data structure

static memory allocation, follows LIFO semantics

- (heap): malloc

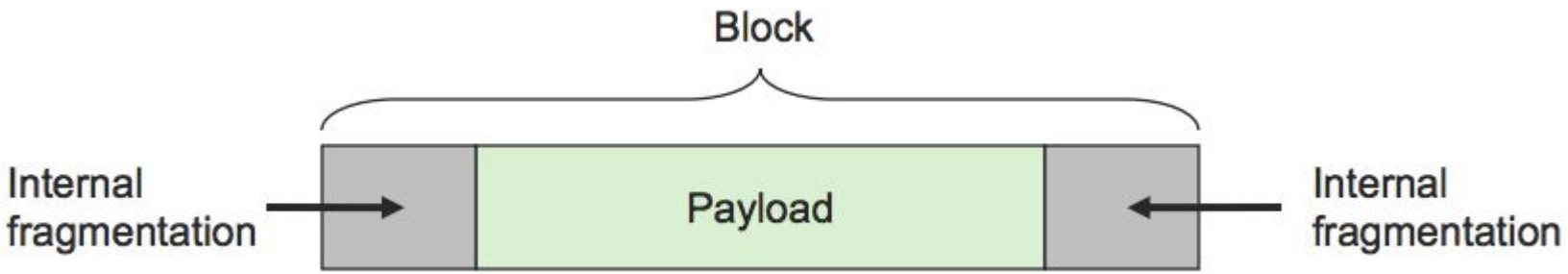
dynamic memory allocation, need to deal with fragmentation





Internal Fragmentation

Payload is smaller than block size



External Fragmentation

There is enough aggregate heap memory, but no single free block is large enough

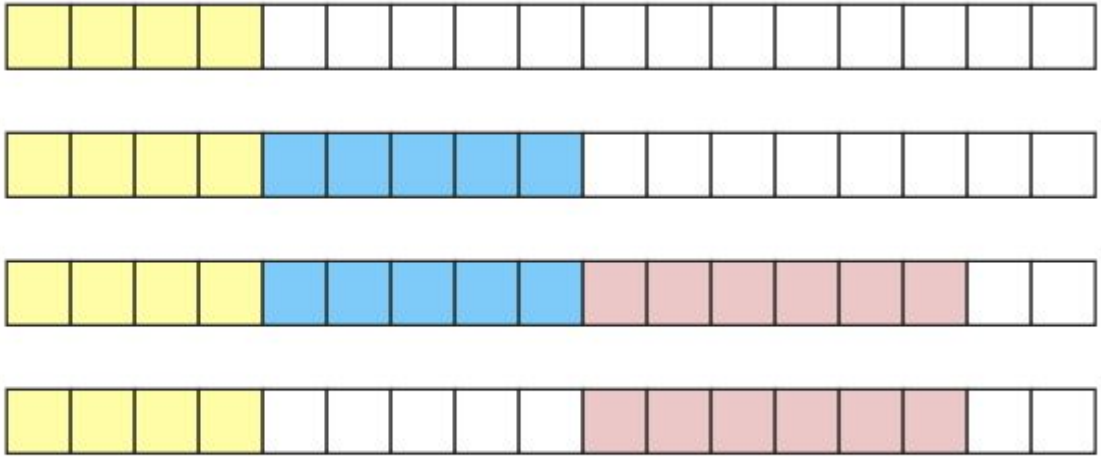
```
p1 = malloc(4)
```

```
p2 = malloc(5)
```

```
p3 = malloc(6)
```

```
free(p2)
```

```
p4 = malloc(6)
```



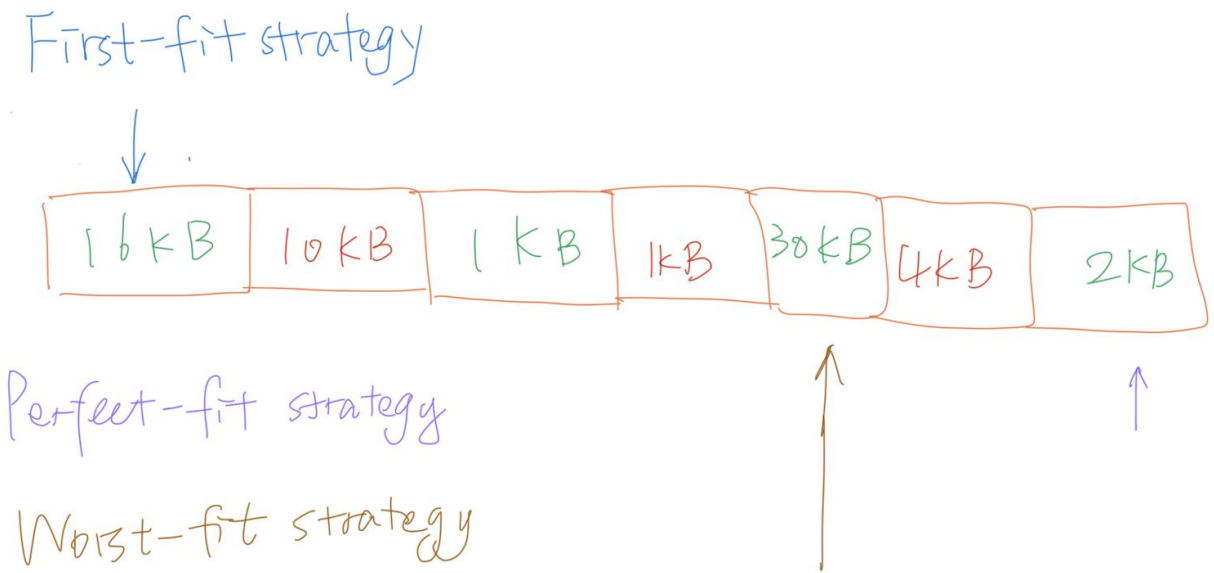
Oops! (what would happen now?)

Placement strategies for dynamic memory allocation

First-fit: pick the first block that fits

Perfect-fit: finds the smallest hole that is of sufficient size

Worst-fit: strategy finds the largest hole that is of sufficient size



Memory leak

If you use malloc to assign a dynamic variable, but not free it. It will cause memory leak.

```
#include <stdlib.h>

void f(void)
{
   int* x = malloc(10 * sizeof(int));
   x[10] = 0;          // problem 1: heap block overrun
}                      // problem 2: memory leak -- x not freed

int main(void)
{
   f();
   return 0;
}
```

Segment fault

If you attempts to

access a memory location

which doesn't exist or is not

allowed to access, a segment

fault will occur.

```
#include <stdio.h>
int main(){
   char *p;
   p = NULL;
   *p = 'x';
   printf("%c", *p);
   return 0;
}
```

```
#include <stdio.h>
 int main(){
      int b = 10;
      printf("%s\n", b);
      return 0;
}
```

Valgrind

Valgrind is a programming tool for **memory debugging**, **memory leak detection**.

Basics of Valgrind

- Official homepage: http://valgrind.org/
- Install on Ubuntu: *sudo apt-get install valgrind*
- Add -g option of gcc to sets up debugging information:

  gcc -g xxx.c -o xxx

- Launch Valgrind by valgrind ./xxx

How to use Valgrind

Show by an example

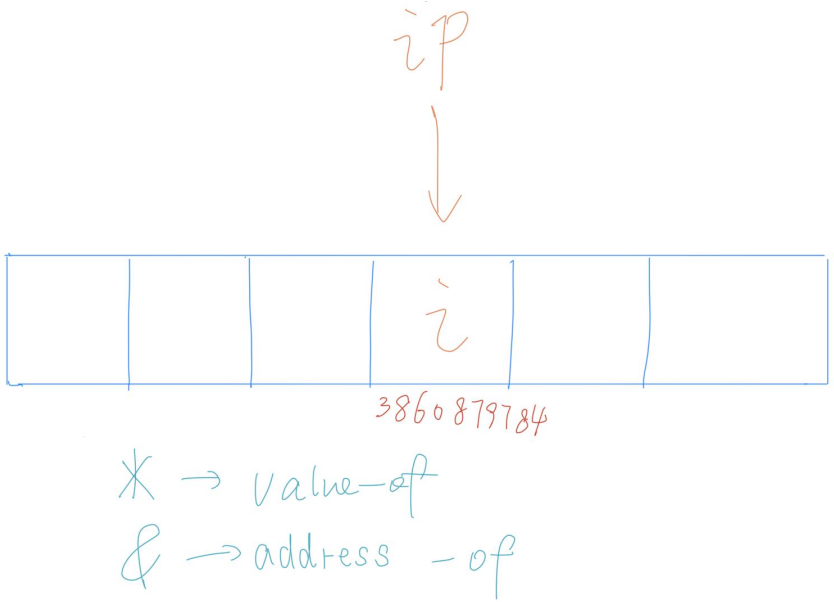
Reference video: https://youtu.be/bb1bTJtgXrI

Homework 2 - Qo Answer

0. Consider:

```
int i = 5;

int *ip = &i;
```



.. what is ip? What is its value?

Homework 2 - Qo Answer

0. Consider:

int i = 5;

int *ip = &i;

.. what is ip? What is its value?

ip is a pointer that stores the address of i

*ip vs ip

```
----------------------------------------
~/2017F/CS 214/Recitation » ./hm2_0
3974719912
----------------------------------------
~/2017F/CS 214/Recitation » ./hm2_0
3989752232
----------------------------------------
~/2017F/CS 214/Recitation » ./hm2_0
3945531816
----------------------------------------
~/2017F/CS 214/Recitation » ./hm2_0
3800770984
----------------------------------------
~/2017F/CS 214/Recitation » ./hm2_0
3866962344
----------------------------------------
~/2017F/CS 214/Recitation » ./hm2_0
3860879784
----------------------------------------
```

Homework 2 - Q1 Answer

1. Write some code that declares two arrays of size 10 that are string literals.

Make a pointer to one of the arrays, cast it to be an int pointer, and print out its value.

Make a new integer, set it equal to the value of your int pointer, then make a pointer to that integer, cast it to be a char pointer, and print out 8 chars.

What happened? Why?

Homework 2 - Q1 Answer

1. Write some code that declares two arrays of size 10 that are string literals.

What happened? Why?

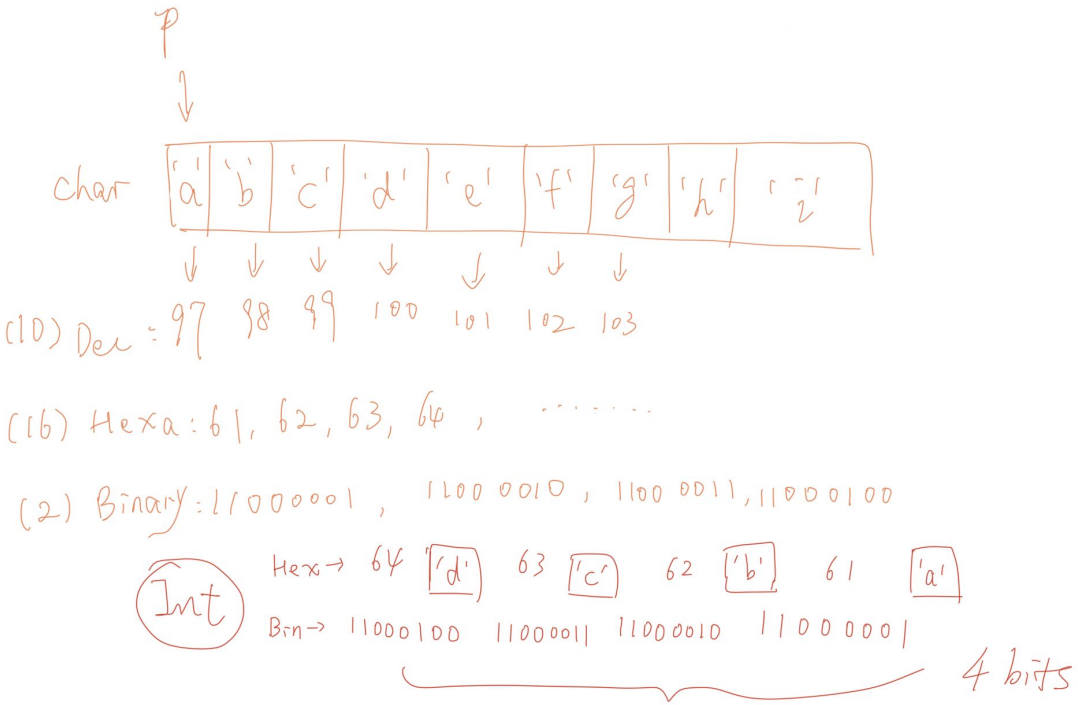
```
~/2017F/CS 214/Recitation » ./hm2_1
1684234849

a
b
c
d




```

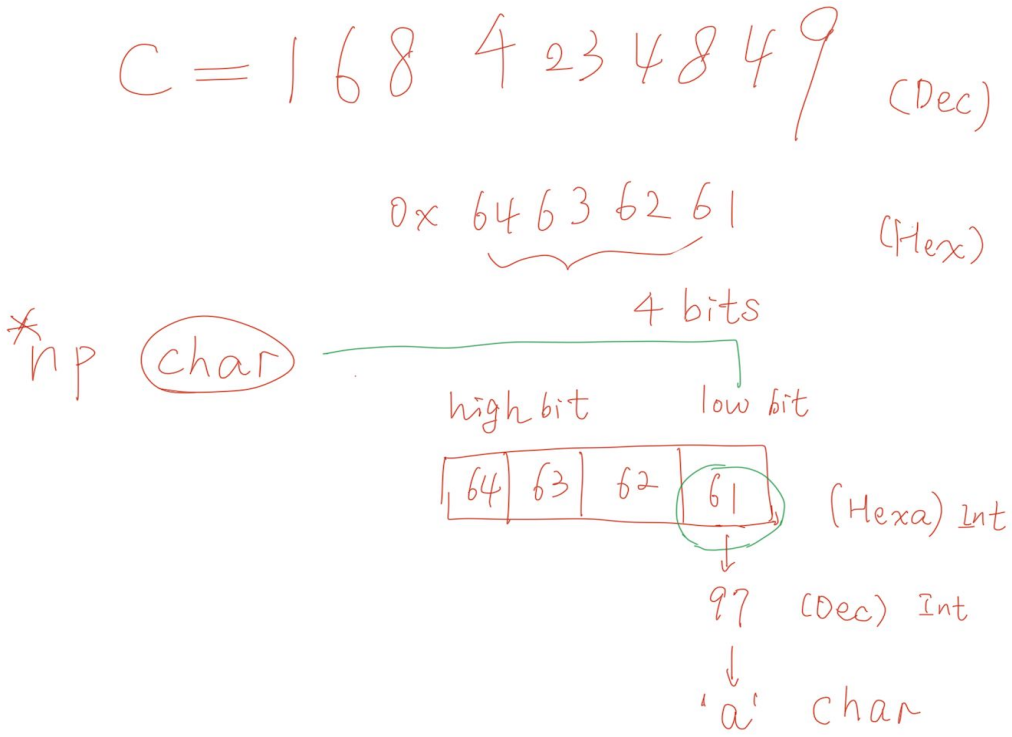
Homework 2 - Q1 Answer



ASCII (1977/1986)

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_	NUL 0000 *0*	SOH 0001 *1*	STX 0002 *2*	ETX 0003 *3*	EOT 0004 *4*	ENQ 0005 *5*	ACK 0006 *6*	BEL 0007 *7*	BS 0008 *8*	HT 0009 *9*	LF 000A *10*	VT 000B *11*	FF 000C *12*	CR 000D *13*	SO 000E *14*	SI 000F *15*
1_	DLE 0010 *16*	DC1 0011 *17*	DC2 0012 *18*	DC3 0013 *19*	DC4 0014 *20*	NAK 0015 *21*	SYN 0016 *22*	ETB 0017 *23*	CAN 0018 *24*	EM 0019 *25*	SUB 001A *26*	ESC 001B *27*	FS 001C *28*	GS 001D *29*	RS 001E *30*	US 001F *31*
2_	SP 0020 *32*	! 0021 *33*	" 0022 *34*	# 0023 *35*	$ 0024 *36*	% 0025 *37*	& 0026 *38*	' 0027 *39*	(0028 *40*	) 0029 *41*	* 002A *42*	+ 002B *43*	, 002C *44*	- 002D *45*	. 002E *46*	/ 002F *47*
3_	0 0030 *48*	1 0031 *49*	2 0032 *50*	3 0033 *51*	4 0034 *52*	5 0035 *53*	6 0036 *54*	7 0037 *55*	8 0038 *56*	9 0039 *57*	: 003A *58*	; 003B *59*	< 003C *60*	= 003D *61*	> 003E *62*	? 003F *63*
4_	@ 0040 *64*	A 0041 *65*	B 0042 *66*	C 0043 *67*	D 0044 *68*	E 0045 *69*	F 0046 *70*	G 0047 *71*	H 0048 *72*	I 0049 *73*	J 004A *74*	K 004B *75*	L 004C *76*	M 004D *77*	N 004E *78*	O 004F *79*
5_	P 0050 *80*	Q 0051 *81*	R 0052 *82*	S 0053 *83*	T 0054 *84*	U 0055 *85*	V 0056 *86*	W 0057 *87*	X 0058 *88*	Y 0059 *89*	Z 005A *90*	[005B *91*	\ 005C *92*	] 005D *93*	^ 005E *94*	_ 005F *95*
6_	` 0060 *96*	a 0061 *97*	b 0062 *98*	c 0063 *99*	d 0064 *100*	e 0065 *101*	f 0066 *102*	g 0067 *103*	h 0068 *104*	i 0069 *105*	j 006A *106*	k 006B *107*	l 006C *108*	m 006D *109*	n 006E *110*	o 006F *111*
7_	p 0070 *112*	q 0071 *113*	r 0072 *114*	s 0073 *115*	t 0074 *116*	u 0075 *117*	v 0076 *118*	w 0077 *119*	x 0078 *120*	y 0079 *121*	z 007A *122*	{ 007B *123*	| 007C *124*	} 007D *125*	~ 007E *126*	DEL 007F *127*

Homework 2 - Q1 Answer



Homework 2 - Q2 Answer

2. Write some code that declares two arrays of size 10 that are string literals.

Create a pointer that points to the beginning of the first array, then in a loop, increment the pointer and print out the char it points to, out to index 20.

What happened? Why?

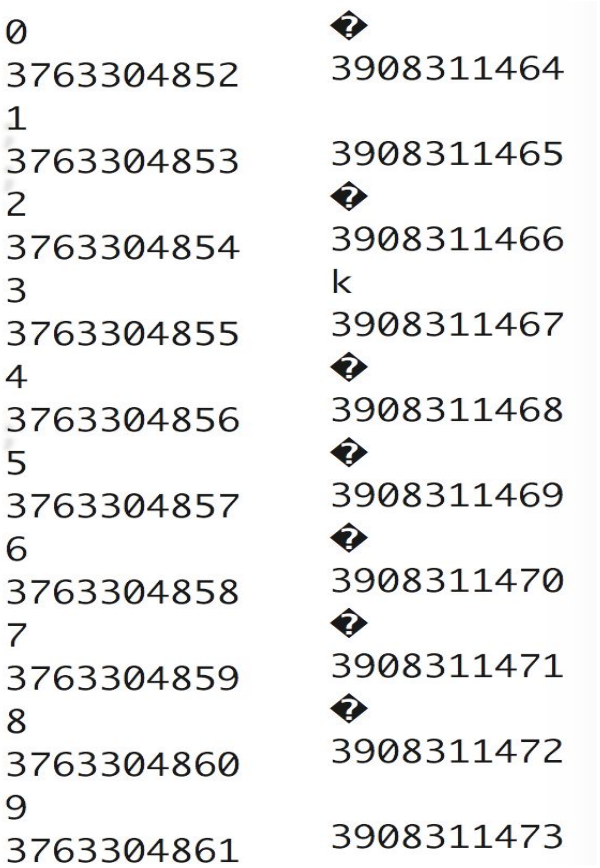
```
0
3763304852
1
3763304853
2
3763304854
3
3763304855
4
3763304856
5
3763304857
6
3763304858
7
3763304859
8
3763304860
9
3763304861
```

```
a
3763304862
b
3763304863
c
3763304864
d
3763304865
e
3763304866
f
3763304867
g
3763304868
h
3763304869
i
3763304870
j
3763304871
```

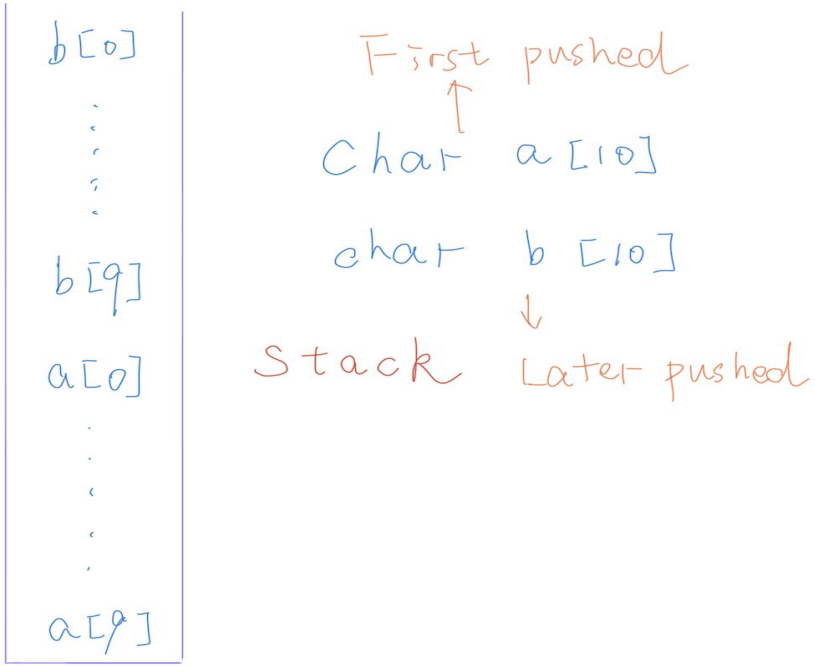
Homework 2 - Q2 Answer

If exchange the order of array a and b, what will happen?

Let's try it!



Homework 2 - Q2 Answer



Homework 3

0. What are the differences between strlen and sizeof a string in C? Why?

1. Write the function: replace(char string[], char from[], char to[])

which finds the string from in the string string and replaces it with the string to. You may assume that from and to are the same length. For example, the code

```
char string[] = "recieve";
replace(string, "ie", "ei");
```

should change string to "receive".

Homework 3 (Cont.)

2.

Write a short program to read two lines of text, and concatenate them using strcat. Since strcat concatenates in-place, you'll have to make sure you have enough memory to hold the concatenated copy. For now, use a char array which is twice as big as either of the arrays you use for reading the two lines. Use strcpy to copy the first string to the destination array, and strcat to append the second one.