This is CS50

Think.
Pair.
Share.

- What are the steps involved in **compilation**?
- What are **arrays**?
- What are **strings**?
- What's the point of **command-line arguments**?

- What are the steps involved in **compilation**?
- What are **arrays**?
- What are **strings**?
- What's the point of **command-line arguments**?
- What makes for good **design**?

```
int main(void)
{
    printf("Hello, world");
}
```

```
...
main:
# @main
    .cfi_startproc
# BB#0:
    push    %rbp
.Ltmp0:
    .cfi_def_cfa_offset 16
.Ltmp1:
    .cfi_offset %rbp, -16
    movq    %rsp, %rbp
.Ltmp2:
    .cfi_def_cfa_register %rbp
```

```
01111111010001010100110001000110
00000010000000010000000100000000
00000000000000000000000000000000
00000000000000000000000000000000
00000001000000000011111000000000
00000001000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
10100000000000010000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
01000000000000000000000000000000
00000000000000000100000000000000
00001010000000000000000100000000
01010101010010001000100111100101
01001000100000111110110000010000
00110001110000001000100111000111
01001000101111110000000000000000
00000000000000000000000000000000
00000000000000001011000000000000
11101000000000000000000000000000
00000000010010001011111100000000
00000000000000000000000000000000
00000000000000000000000001001000
```

```
$ clang
$ clang hello.c
$ clang -o hello hello.c
$ make hello
```

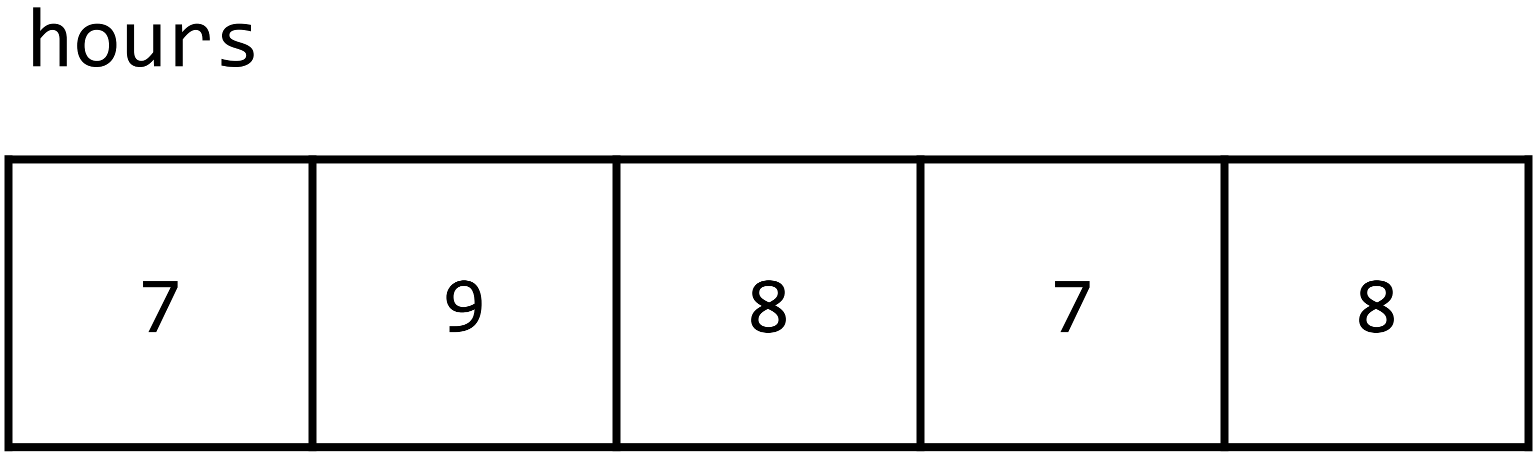
Arrays

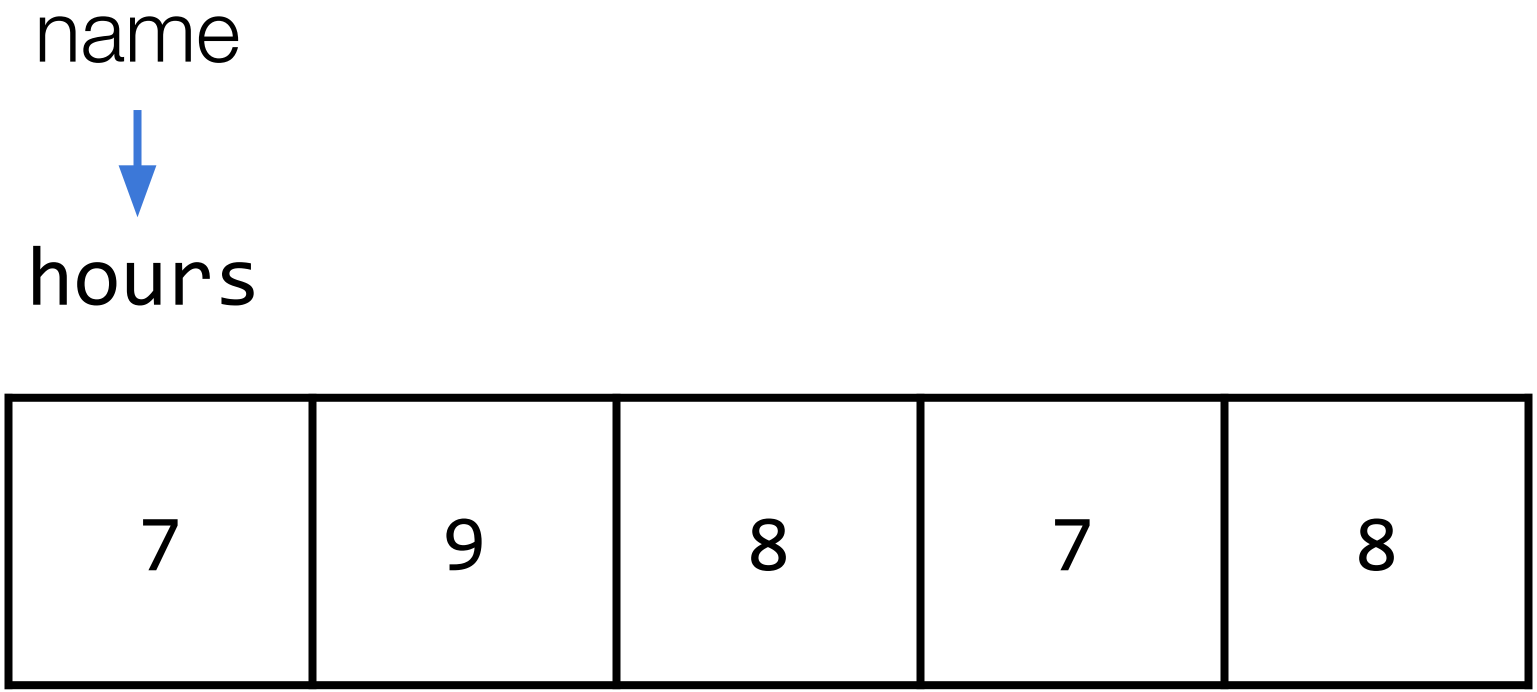
good morning ☀️ it's a new day!
how was last night? 👀

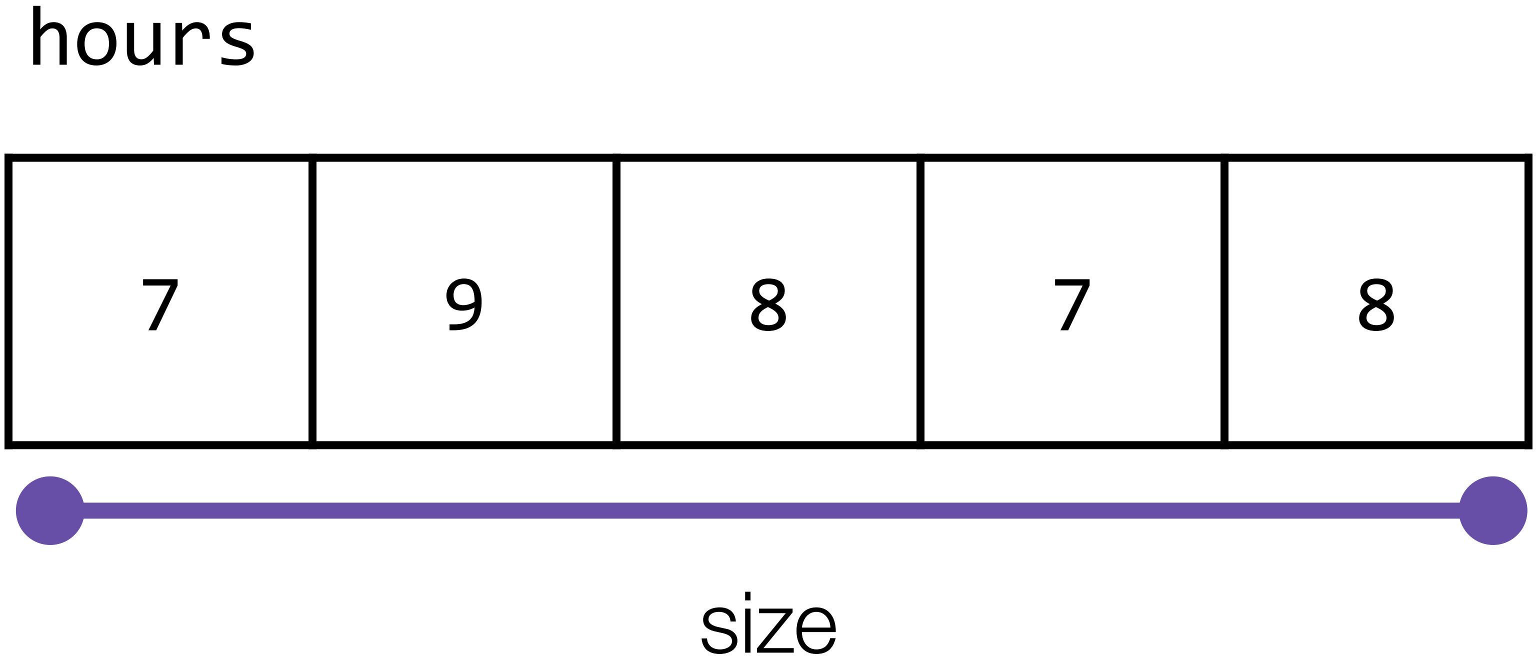
how many hours did you sleep last night? *

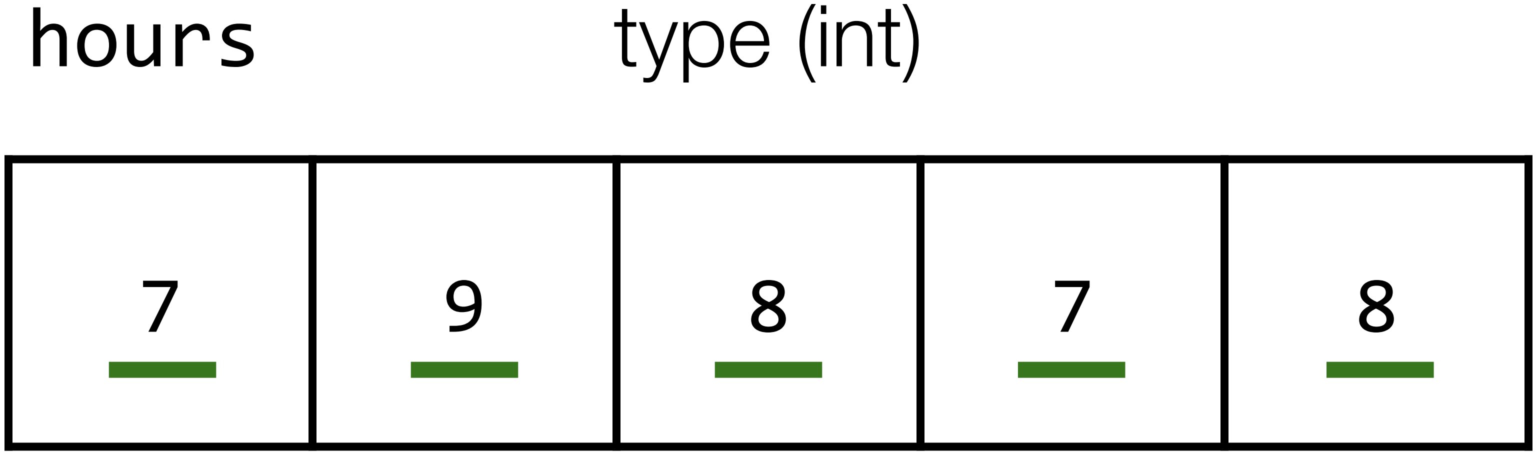
7.5

```
int hours_1 = 7;
int hours_2 = 9;
int hours_3 = 8;
int hours_4 = 7;
int hours_5 = 8;
```

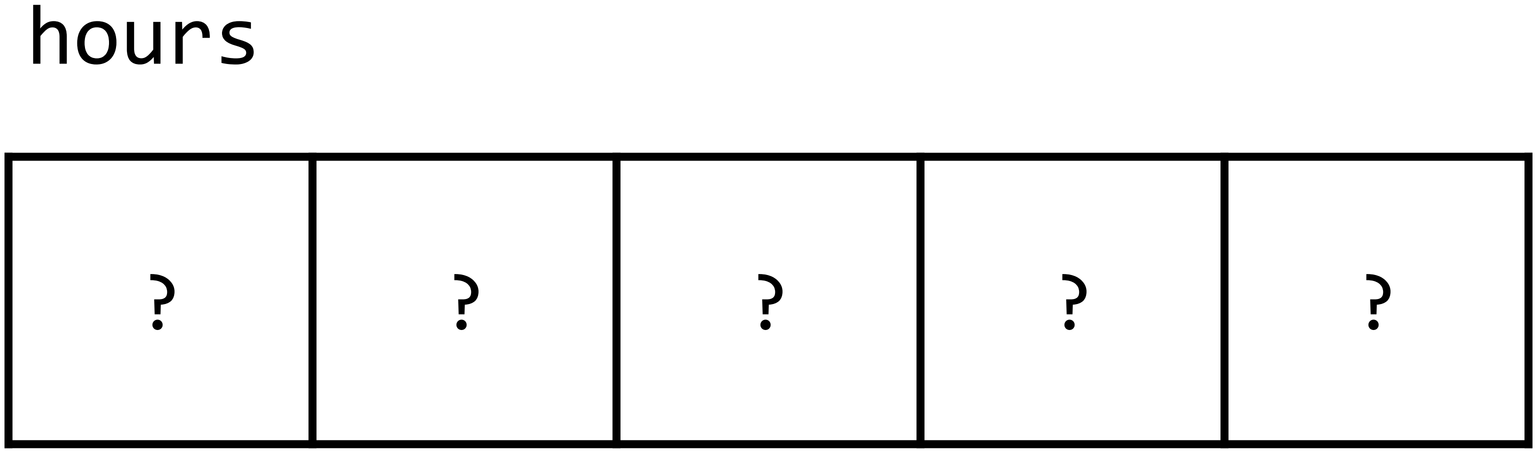

hours
7
9
8
7
8


name
hours
7
9
8
7
8


hours
7
9
8
7
8
size

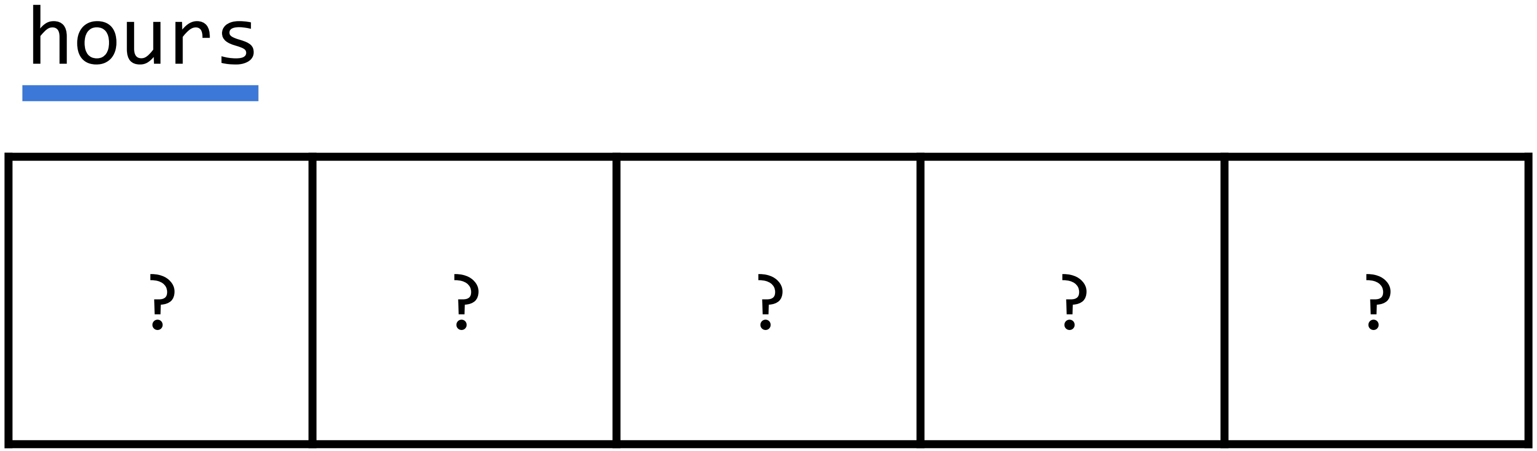

hours
type (int)
7
9
8
7
8

```
int hours[5];
```



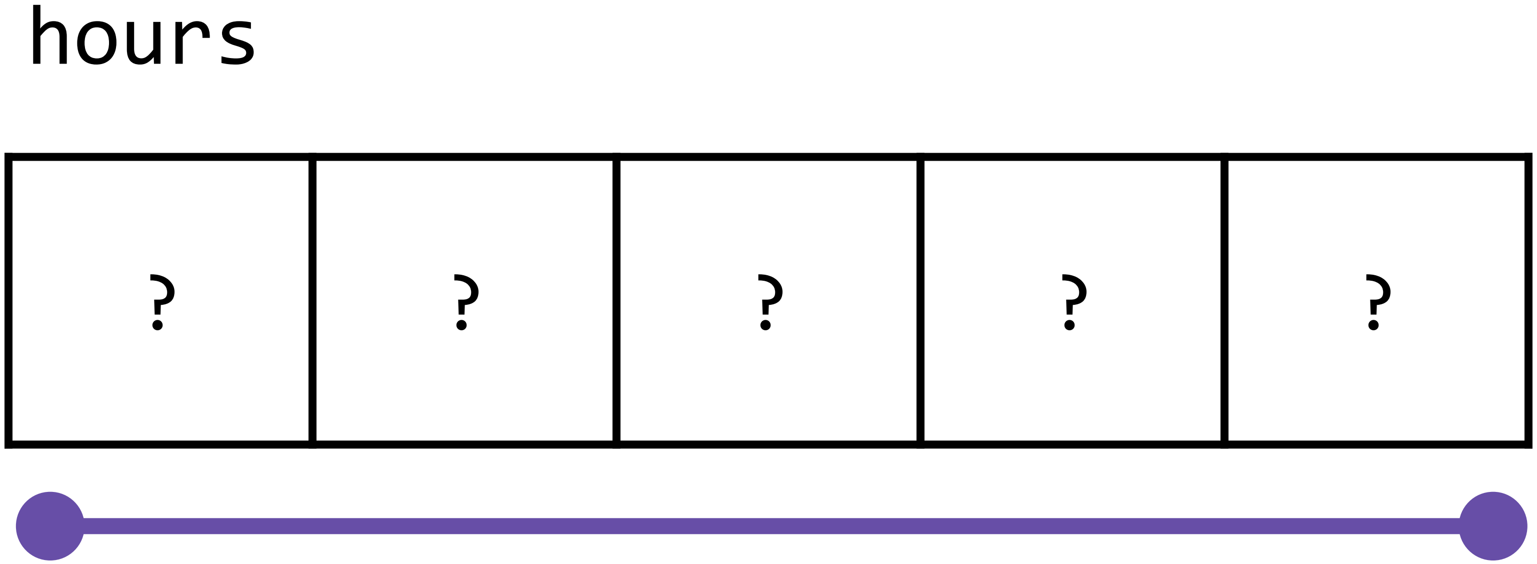
name

```
int hours[5];
```



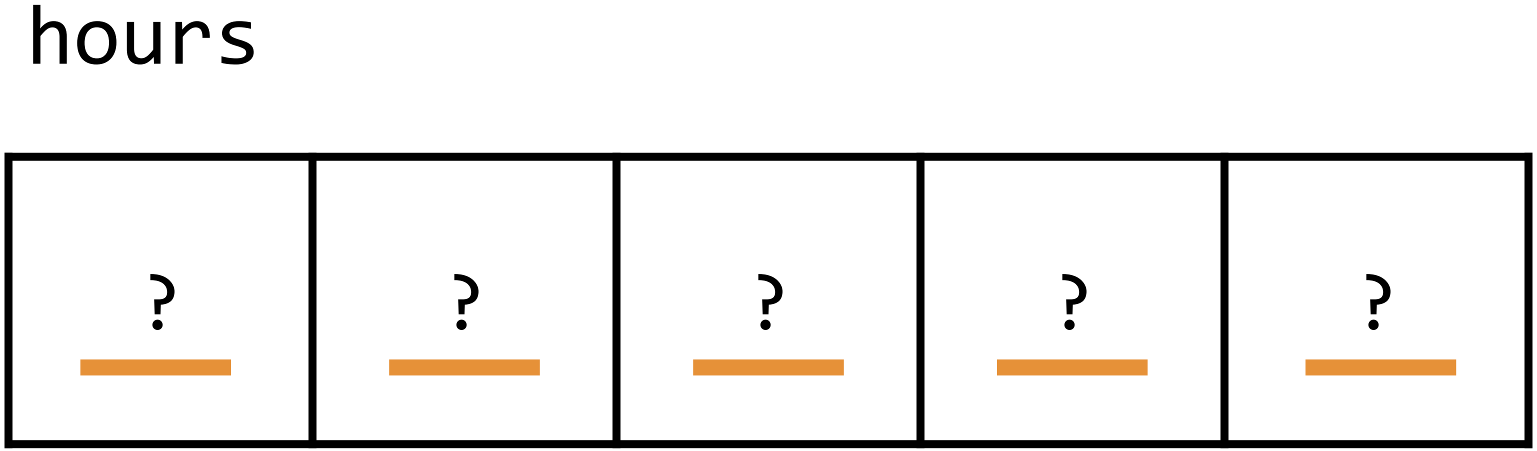
size

```
int hours[5];
```

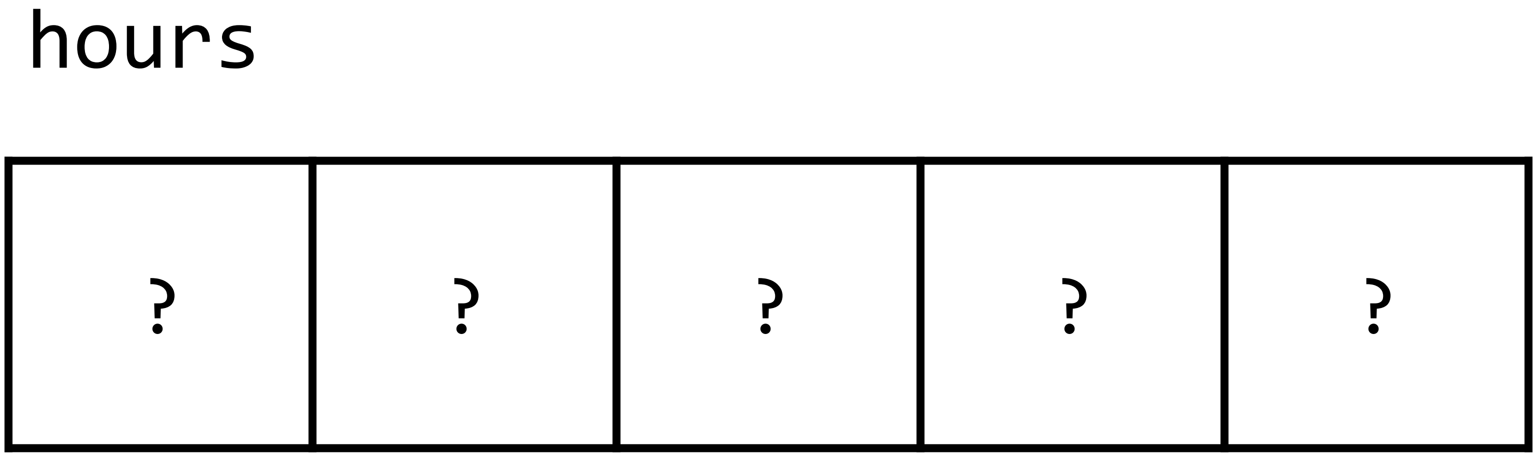


type

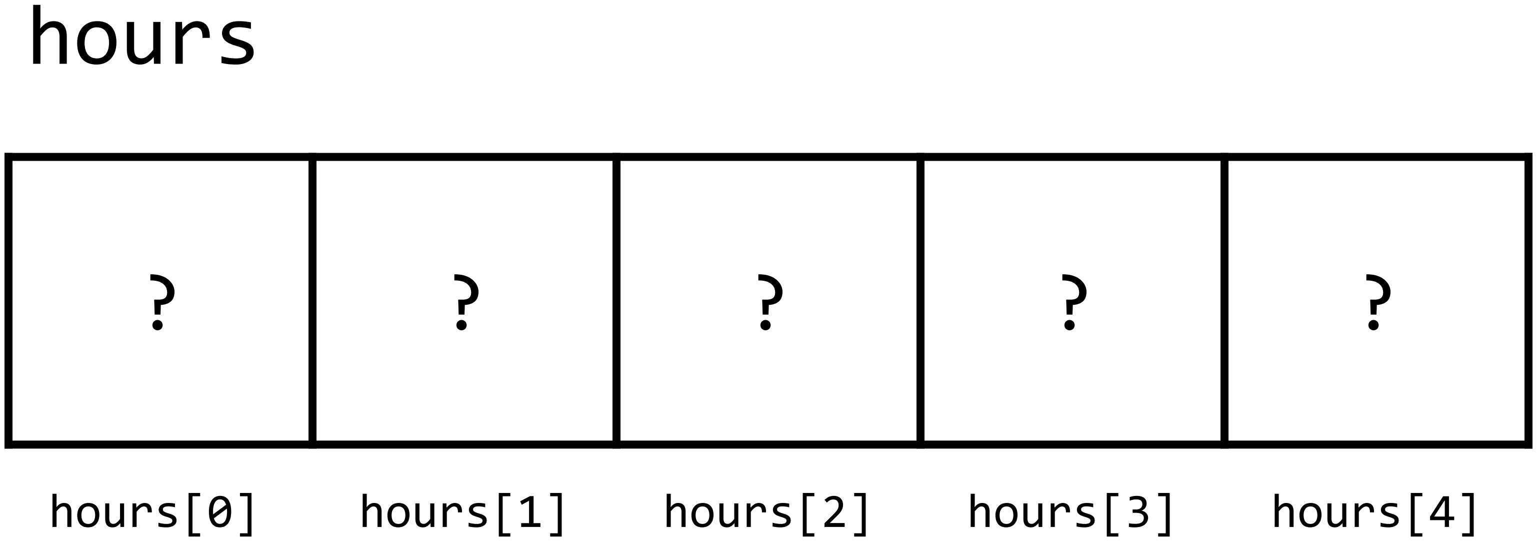
```
int hours[5];
```



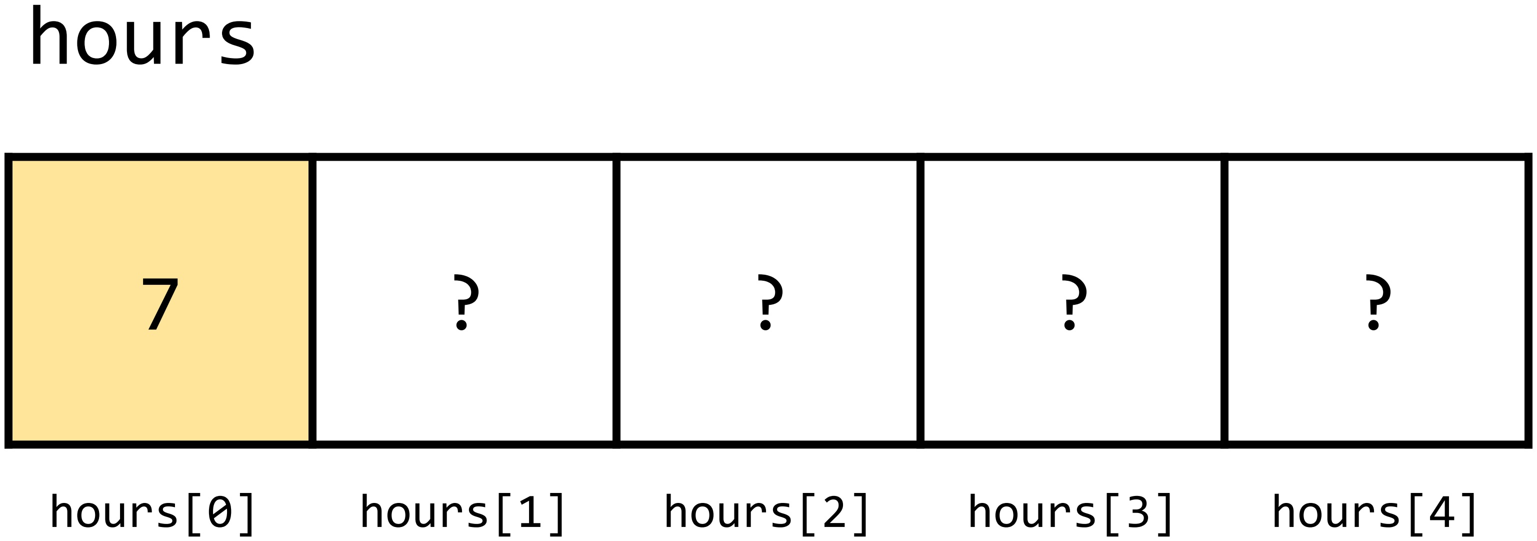
```
int hours[5];
```



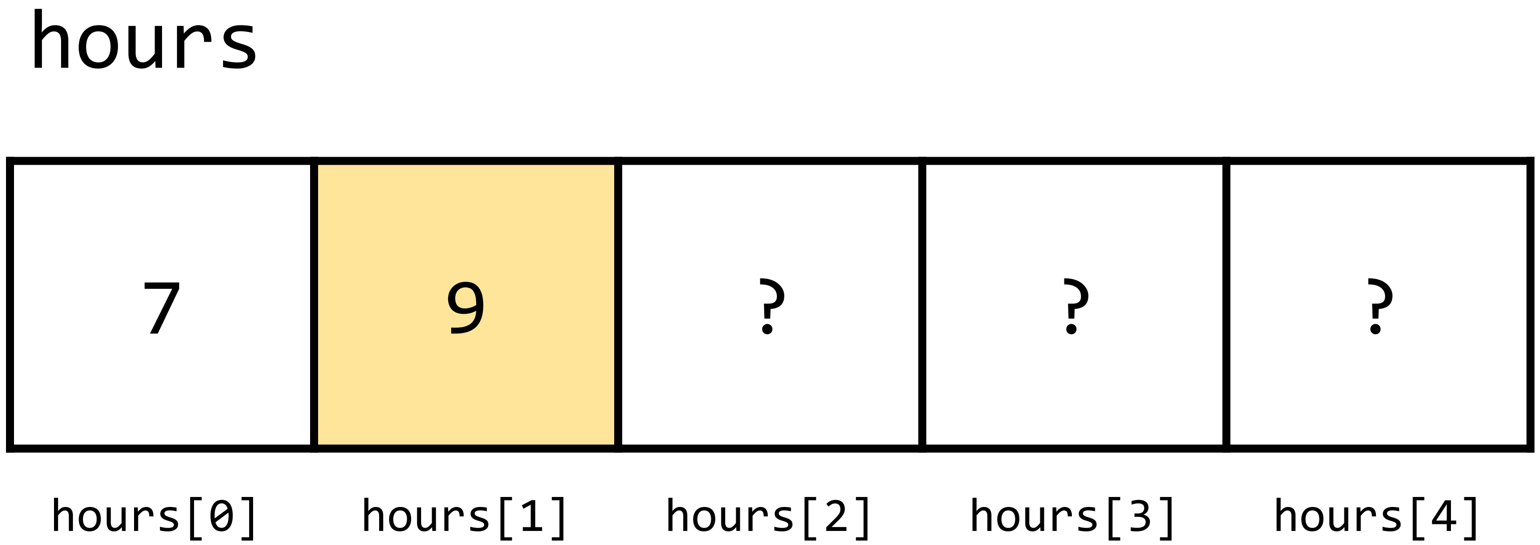
```
int hours[5];
```



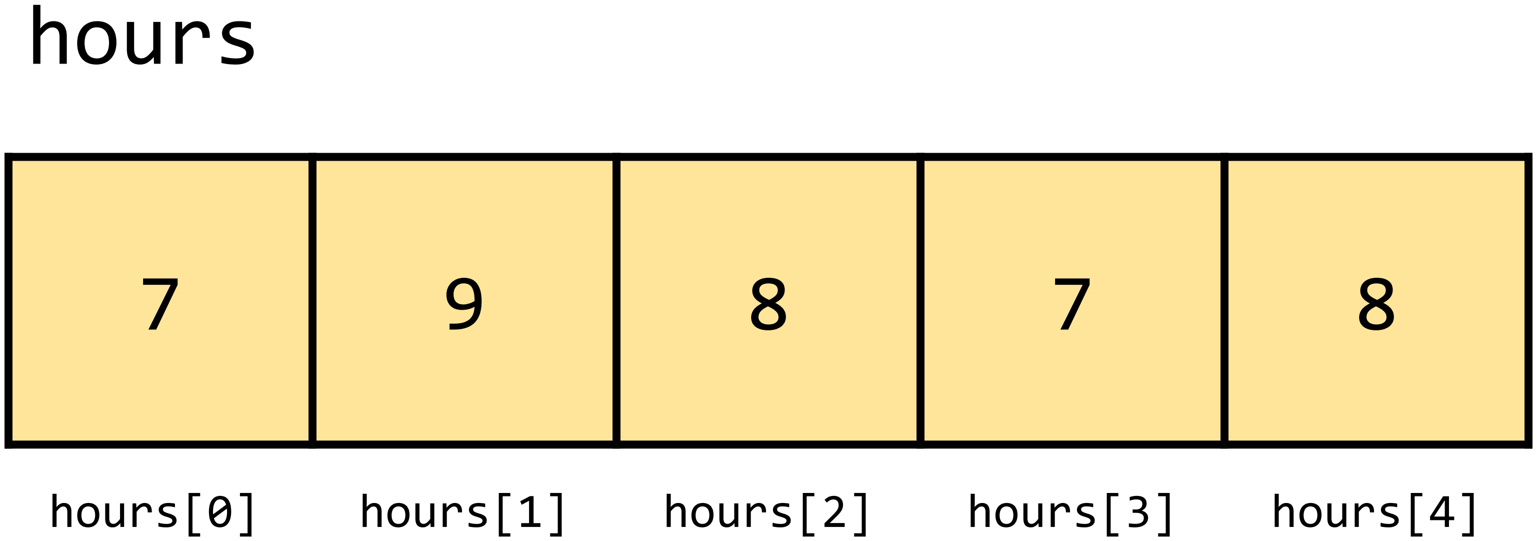
```
int hours[5];
hours[0] = 7;
```



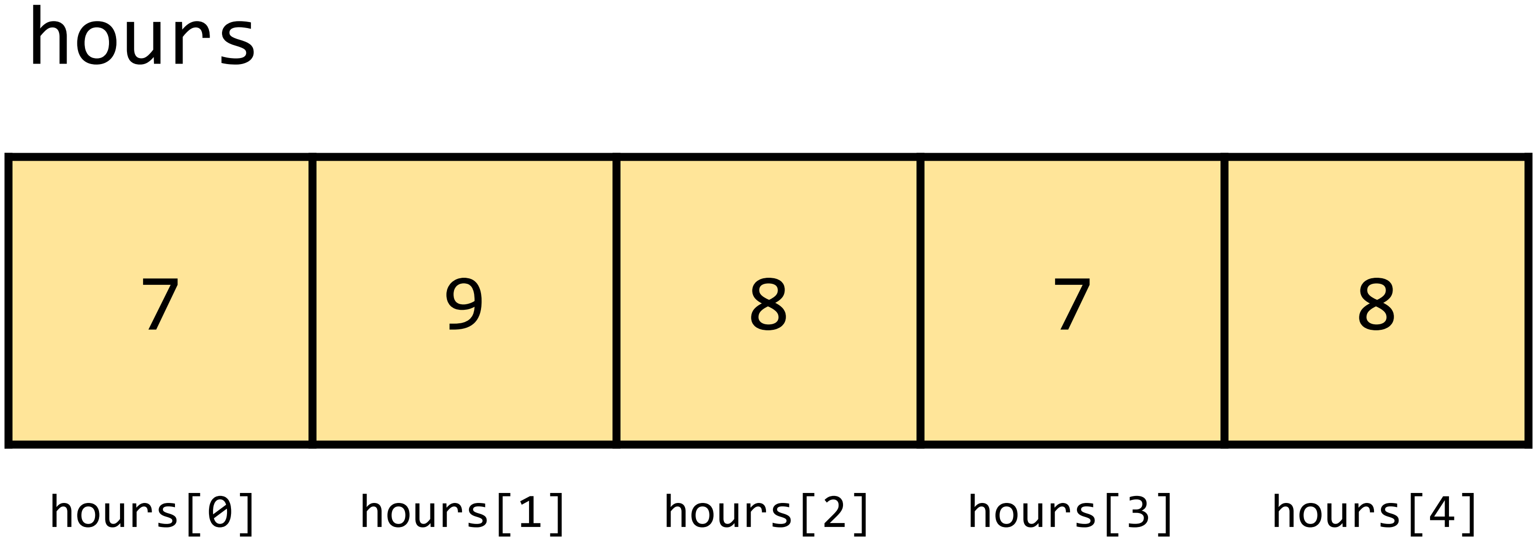
```
int hours[5];
hours[0] = 7;
hours[1] = 9;
```



```
int hours[5] = {7, 9, 8, 7, 8};
```



```
int hours[] = {7, 9, 8, 7, 8};
```



```c
int hours[] = {7, 9, 8, 7, 8};

for (int i = 0; i < 5; i++)
{
    printf("%i\n", hours[i]);
}
```

```
int hours[] = {7, 9, 8, 7, 8};

for (int i = 0; i < 5; i++)
{
    printf("%i\n", hours[i]);
}
```

Powers of 2

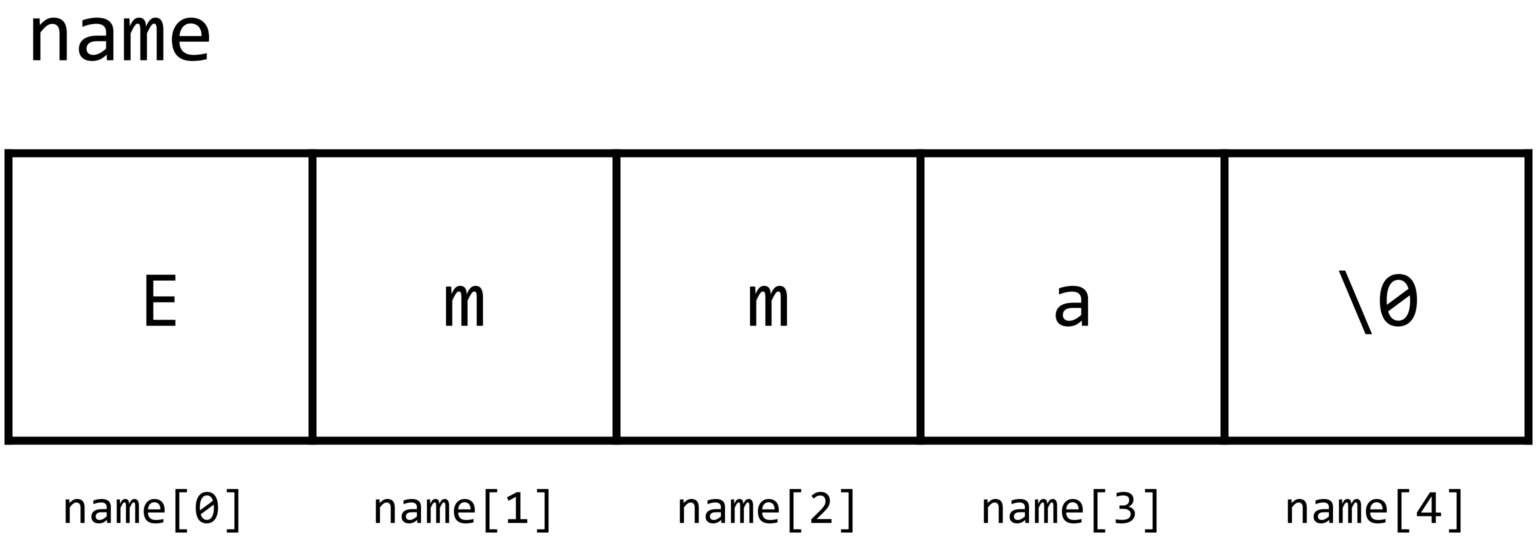
Create a program that prompts the user for a size, **n**. Dynamically create an array of that size, where each element is 2 times the previous one.

Start the array at 1.

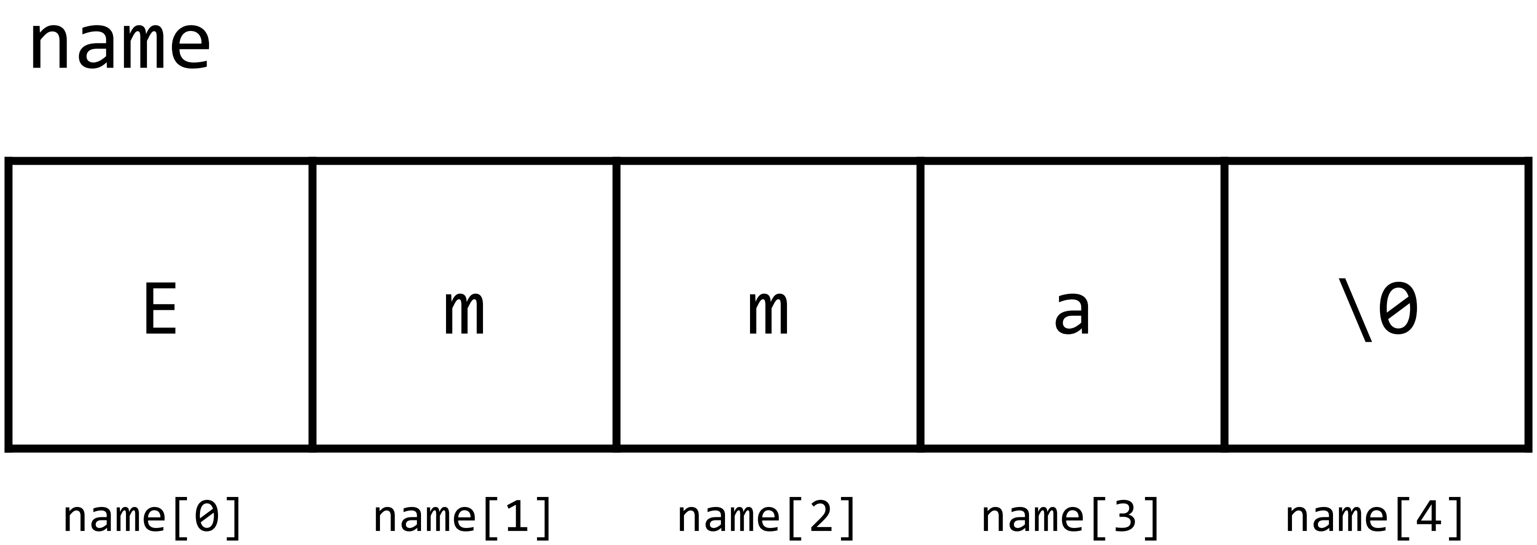
Print the array, integer by integer.

Strings

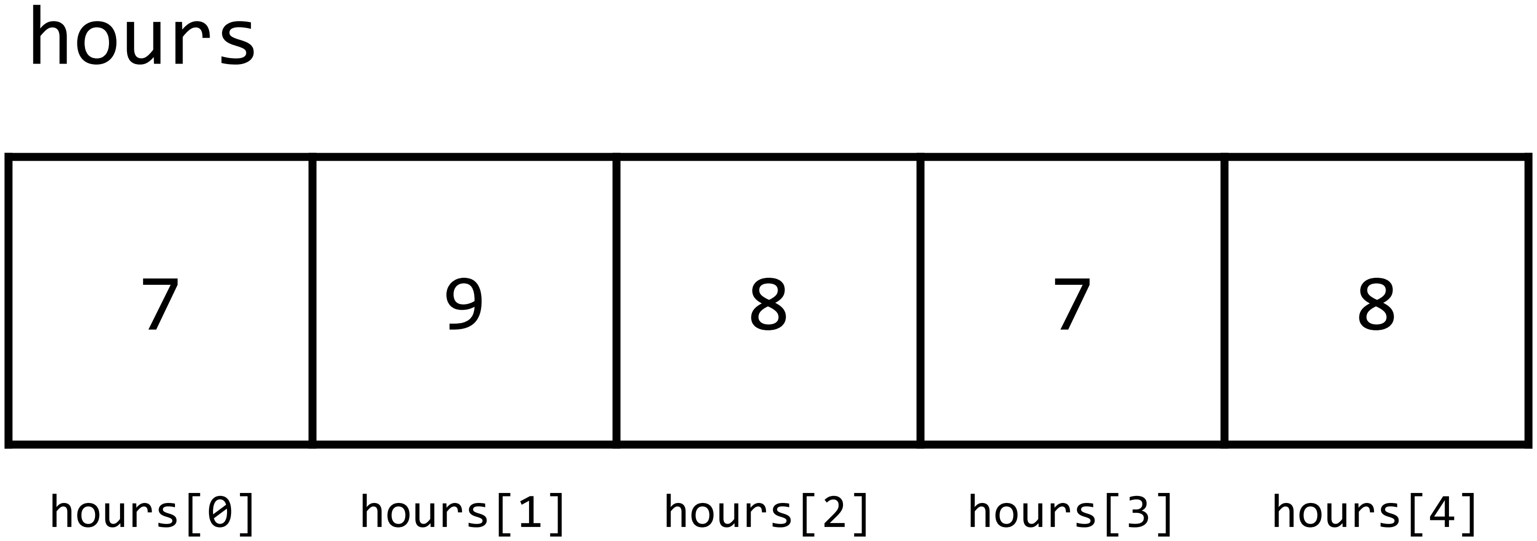
```
string name = "Emma";
```



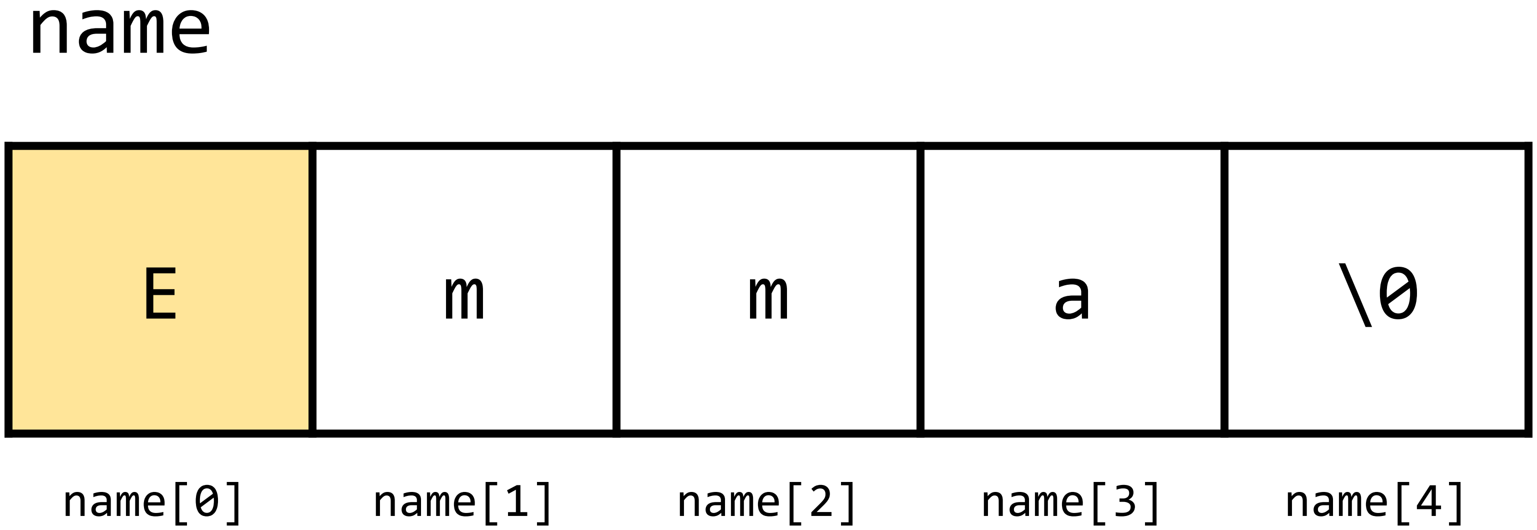
```c
char name[] = {'E', 'm', 'm', 'a', '\0'};
```



```
int hours[] = {7, 9, 8, 7, 8};
```

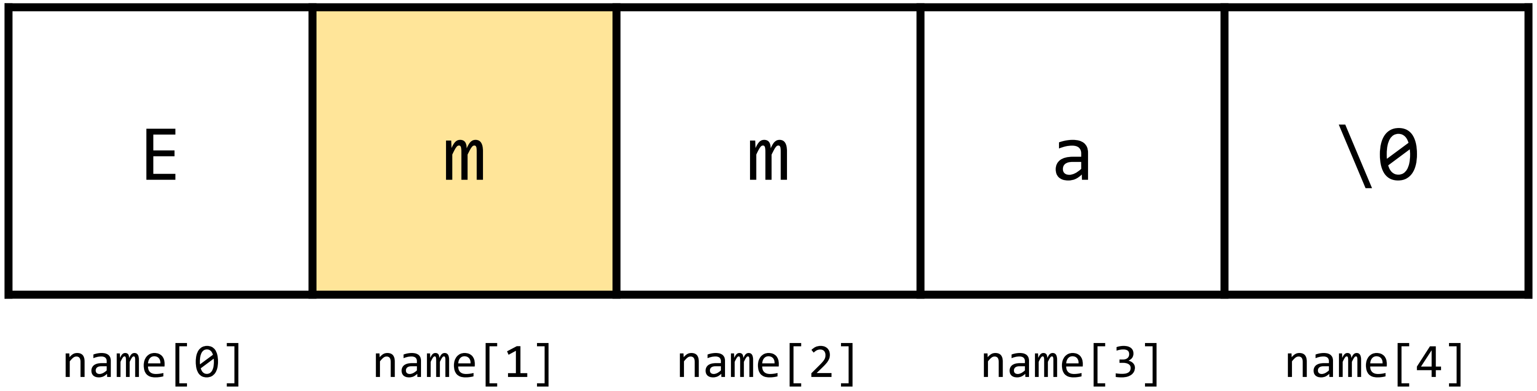


```
name[0];
```



```
name[1];
```

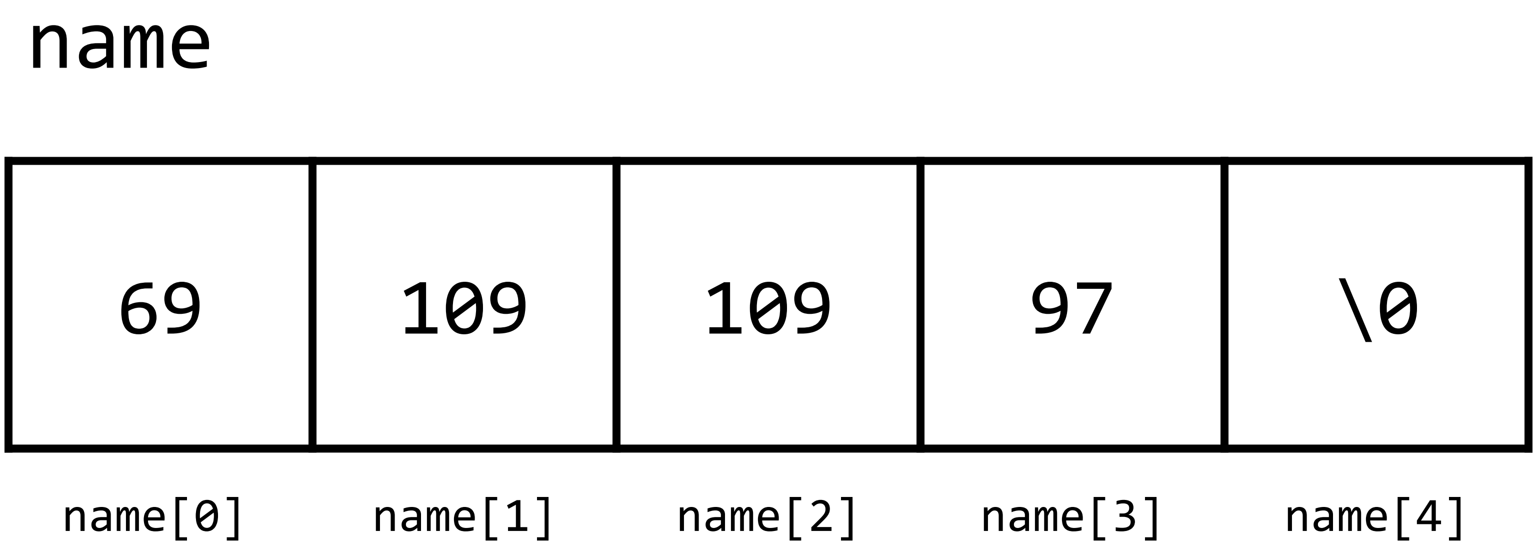
```
name
```



A	**B**	**C**	...	**Z**
65	66	67	...	90

a	**b**	**c**	...	**z**
97	98	99	...	122

```
string name = "Emma";
```



Alphabetical Exercise

Check if a lowercase string's characters are in alphabetical order. If yes, print "Yes". If no, print "No".

asciichart.com

Command-line Arguments

```
$ clang
$ clang mario.c
$ clang -o mario mario.c
$ make mario
```

```
int calculate_quarters(int cents)
{
    ...
}
```

Function argument(s)

```
int calculate_quarters(int cents)
{
    ...
}
```

```
int main(void)
{
    ...
}
```

```
int main(int argc, string argv[])
{
    ...
}
```

```
int main(int argc, string argv[])
{
    ...
}
```

$$\texttt{\$}\ \underbrace{\texttt{make}}_{\text{argv[0]}}\ \underbrace{\texttt{mario}}_{\text{argv[1]}}$$

```
$ ./initials Carter Zenke
```

$$\texttt{\$}\ \underbrace{\texttt{./initials}}_{\text{argv[0]}}\ \underbrace{\texttt{Carter}}_{\text{argv[1]}}\ \underbrace{\texttt{Zenke}}_{\text{argv[2]}}$$

```
$ ./initials Carter Zenke
```

argv[1][0] argv[2][0]

Initials

Given a name as a set of command-line arguments, print the initials of that name to the terminal.

This was CS50