Lab #6- One-dimensional arrays and strings

TA: Alliyya Mo

Today's Lab

- Announcements
- Reviewing
  - STYLE MATTERS → Style Guidelines
  - for loops
  - arrays
  - Strings
- Lab Exercises
- Homework activity DUE Tomorrow @ 11:55 pm
  - Late submissions == 0 || Doesn't compile == 0
- Second Hr:
  - A2 questions & Extra help
- Reminder:
  - Do textbook things
  - A2

Review from last few weeks

- Algorithm Format
- Basic C program (#include <stdio> … return (0); };)
- How to get basic input and output (printf & scanf)
- How to compile (gcc file.c -Wall -std=c99 -o runMe)
- How to to run your code (./runMe)
- Basic terminal maneuvering
- How to submit to moodle
- Conditional Statements
  - If statements (if..else if .. else)
  - Switch
- Loops
  - For
  - While/Do While
- User defined functions with Pass by value

Office Hours

Where: Reynolds 3324

When:

- Monday 4:30 - 6:30pm - Arslan
- Wednesday 2:00 - 4:30pm - Devon (in Thornborough 3401)
- Wednesday 4:30 - 6:30pm - Dillon
- Thursday 2:30pm - 4:30pm
  - Braden
  - Lindsay
- Friday 3:30 - 5:30pm
  - Thomas
  - Lucas

Code Clinics

Code clinics are for guided practice in writing code.

For Tuesday / Thursday sections:

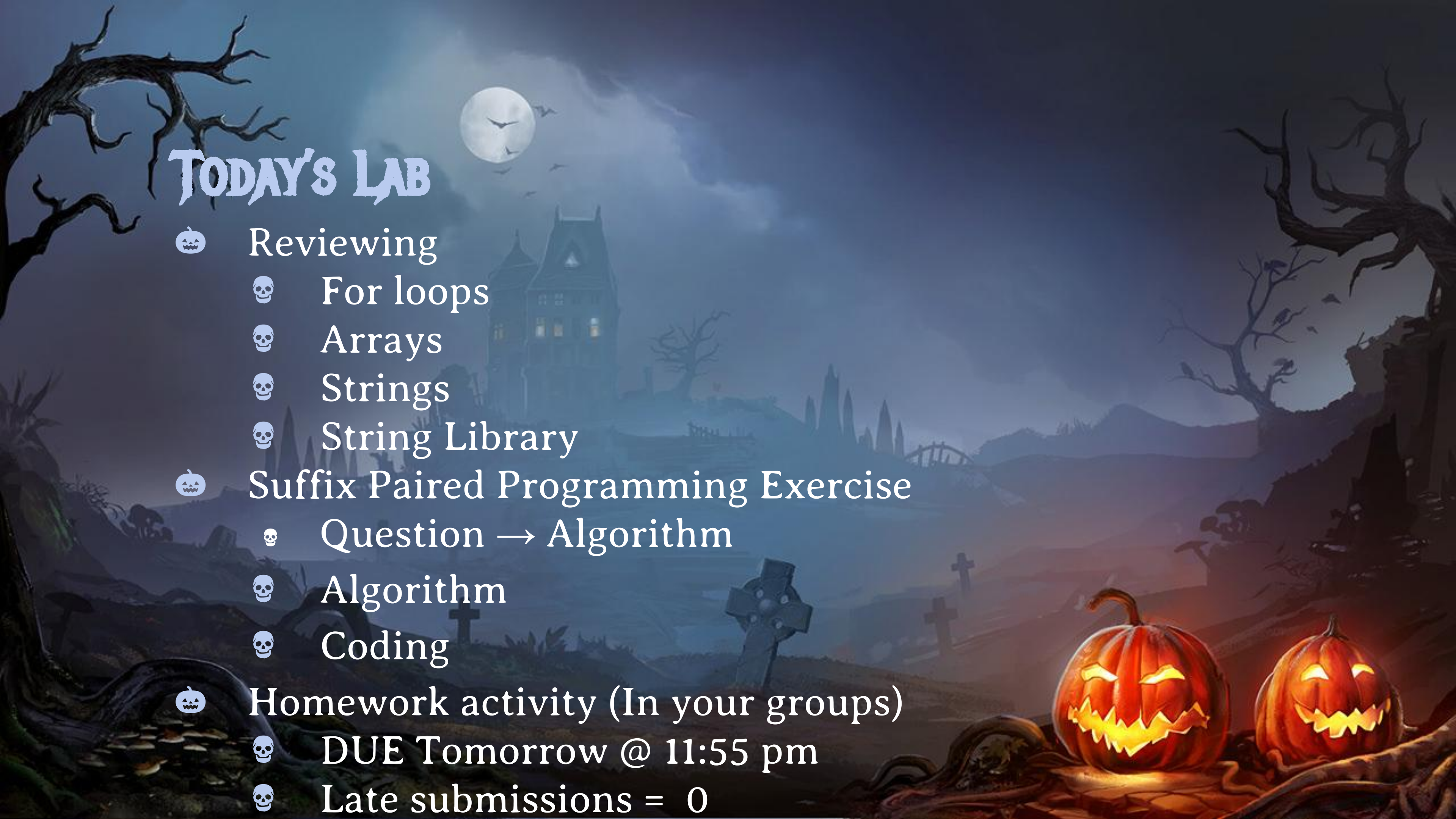
Wednesday October 31st - 4:45 - 5:45pm

Wednesday November 7th - 4:45 - 5:45pm

For Monday / Wednesday section:

Thursday November 1st - 2:45 - 3:45pm

Thursday November 8th - 2:45 - 3:45pm

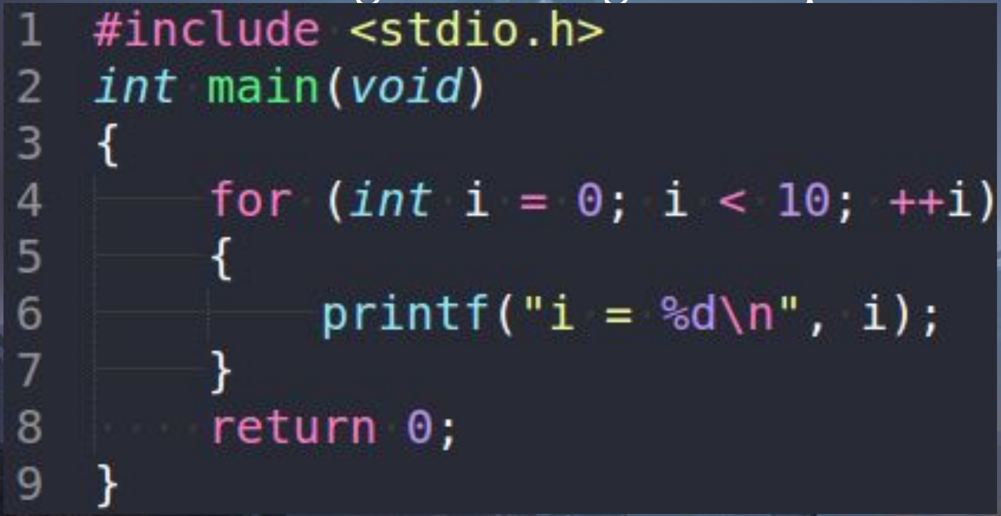


Today's Lab

- Reviewing
  - For loops
  - Arrays
  - Strings
  - String Library
- Suffix Paired Programming Exercise
  - Question → Algorithm
  - Algorithm
  - Coding
- Homework activity (In your groups)
  - DUE Tomorrow @ 11:55 pm
  - Late submissions = 0

For Loops

Understanding and using for loops is vital to using arrays.



```c
#include <stdio.h>
int main(void)
{
	for (int i = 0; i < 10; ++i)
	{
		printf("i = %d\n", i);
	}
	return 0;
}
```

Iterate a set number of times

Used when the number of iterations is known before the loop begins

Mini Lab Exercise

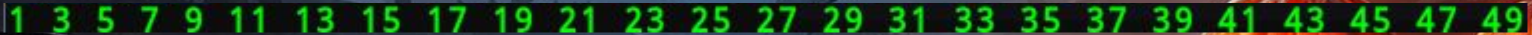
Create a For Loop that iterates through all the odd numbers from between 1-50

Lab Setup Reminder

Open up terminal: type the following

```
cd
mkdir lab6
cd lab6
nano oddNum.c
```

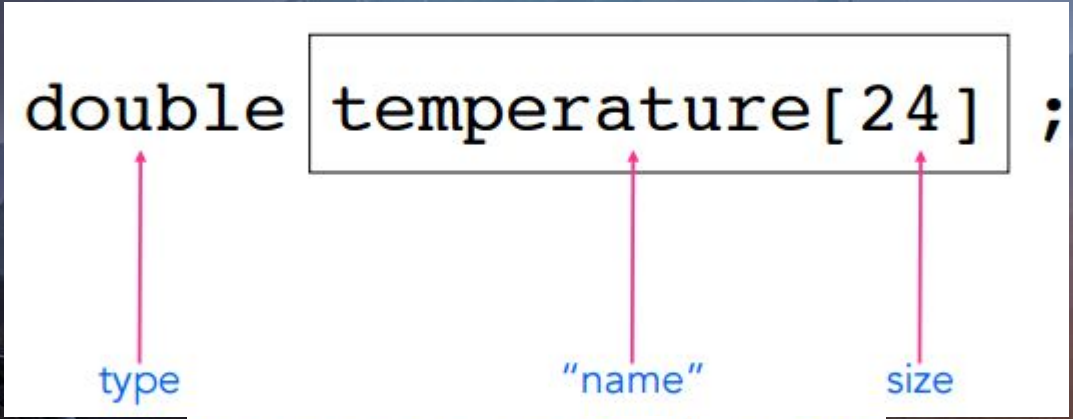
Expected output:



```
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
```

Arrays

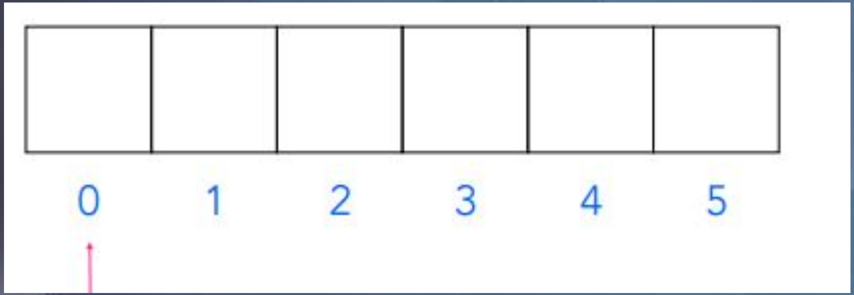
An array is a consecutive series of variables that share one variable name.



```
type name[size];
```

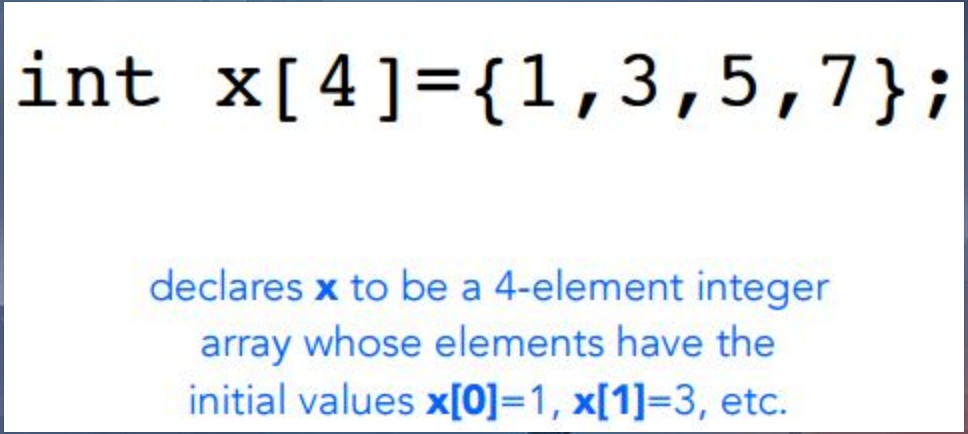
Arrays start at 0

int numArray[6]   //6 elements



Indexed from 0-5

Initializing a Static Array



```
int x[4]={1,3,5,7};
```

declares **x** to be a 4-element integer array whose elements have the initial values **x[0]**=1, **x[1]**=3, etc.

Array elements

loop indexes as
array subscripts

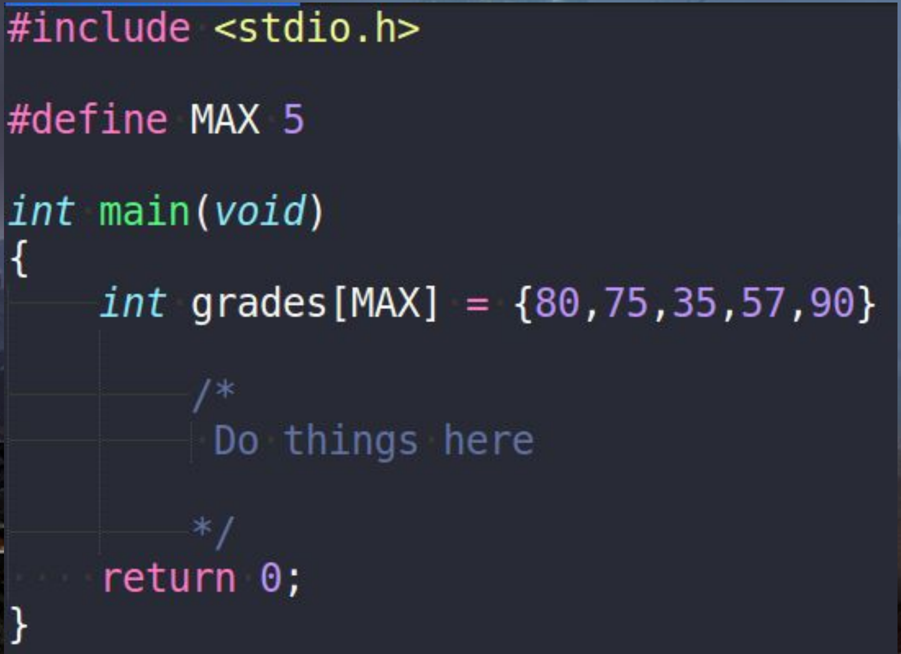
```
int i, z[100];

for (i=0; i<100; i=i+1)
    z[i] = i;
```

for loop to process
elements in an array

each time the value
of a control variable
is incremented, the
next array element is
selected.

Simple Static Int array (Mini Lab exercise II)

Extend the following code to include the calculation of the avg of grades



```c
#include <stdio.h>

#define MAX 5

int main(void)
{
    int grades[MAX] = {80,75,35,57,90}

            /*
            Do things here

            */
    return 0;
}
```

How many elements?

So far, we’ve been allocating arrays statically. This means that we cannot increase or reduce the size of our array after it has been declared.

It is possible to create dynamic arrays, but this won’t be covered until CIS*2500.

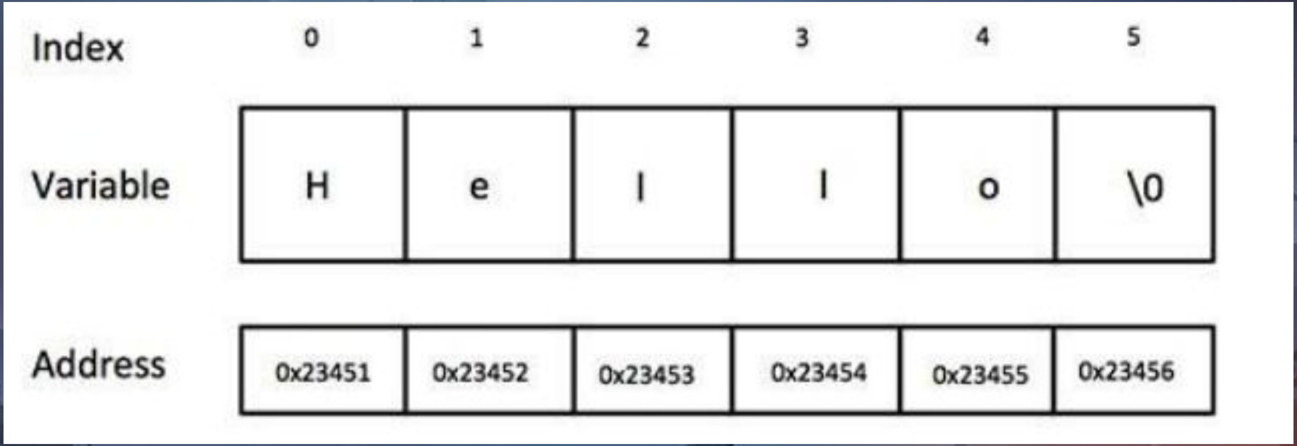
As a rule of thumb, when using static arrays is it better to have too much room as opposed to too little.

Probably not going to want to do arrayName[9999999]

Strings!

- strings are just one dimensional arrays of characters with a special ending character
  - the null terminator '\0'
- The final character of the array must be a null terminator in order for it to be a string to perform string based functions upon it
- You must size your arrays keeping in mind of this null terminator

CHAR WORD[] = "HELLO";



Index	0	1	2	3	4	5
Variable	H	e	l	l	o	\0
Address	0x23451	0x23452	0x23453	0x23454	0x23455	0x23456

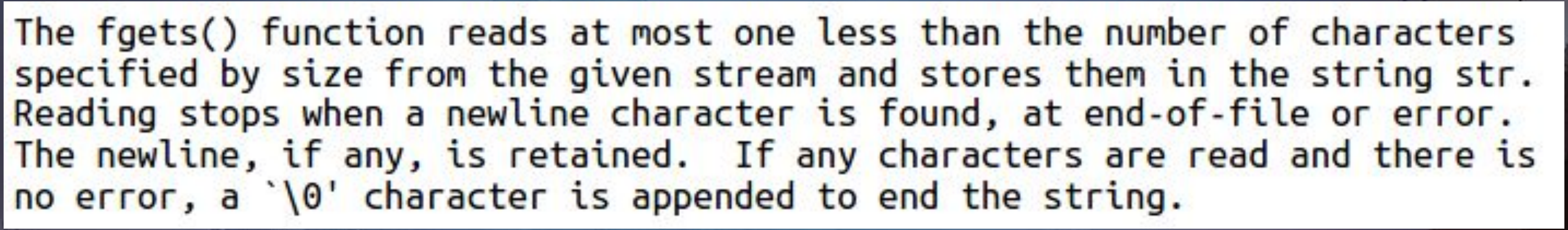
Null Terminator ('\0')

- The null terminator is a special character that in a character indicates the end of the string
- For a character array to be considered a valid string it must have a null terminator

fgets

The answer to life the universe and everything

char *fgets(char *s, int size, FILE *stream)



```
The fgets() function reads at most one less than the number of characters
specified by size from the given stream and stores them in the string str.
Reading stops when a newline character is found, at end-of-file or error.
The newline, if any, is retained.  If any characters are read and there is
no error, a `\0' character is appended to end the string.
```

It return NULL on failure

Standard Streams include stdin, stdout,stderr

FGETS

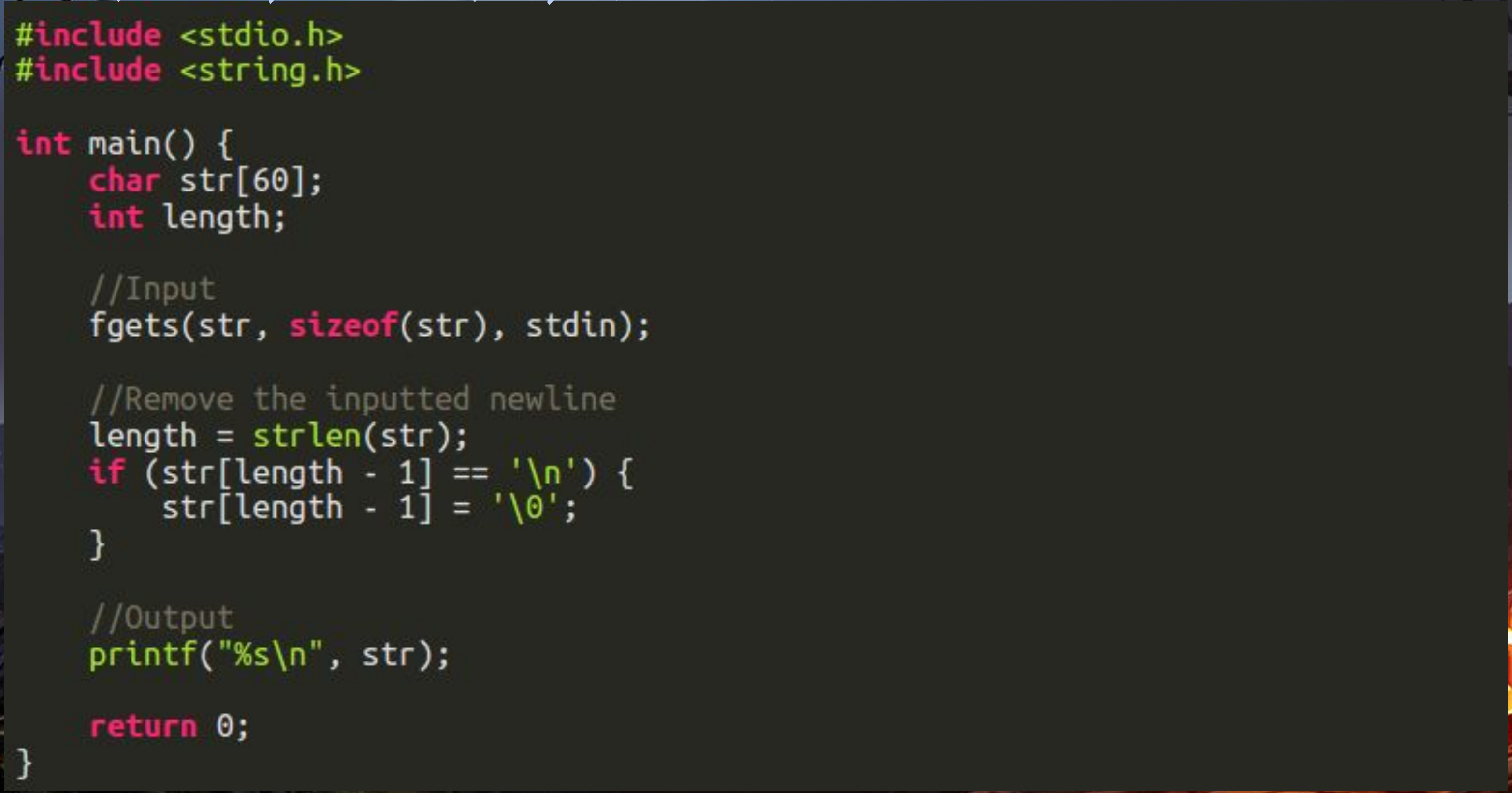
- The format of fgets is:

  char *fgets(char *string, int size, FILE *stream)

- str – The pointer to an array of characters where the string read is stored.
- max_chars – The maximum number of characters to be read (including the null terminator).
- input_stream – The pointer to a FILE pointer that identifies the stream where characters are read from.

On success, fgets returns the same str parameter. Otherwise, if the EOF character is encountered and no characters have been read, the contents of str remain unchanged and a null pointer is returned

FGETS: COMES WITH A NEWLINE



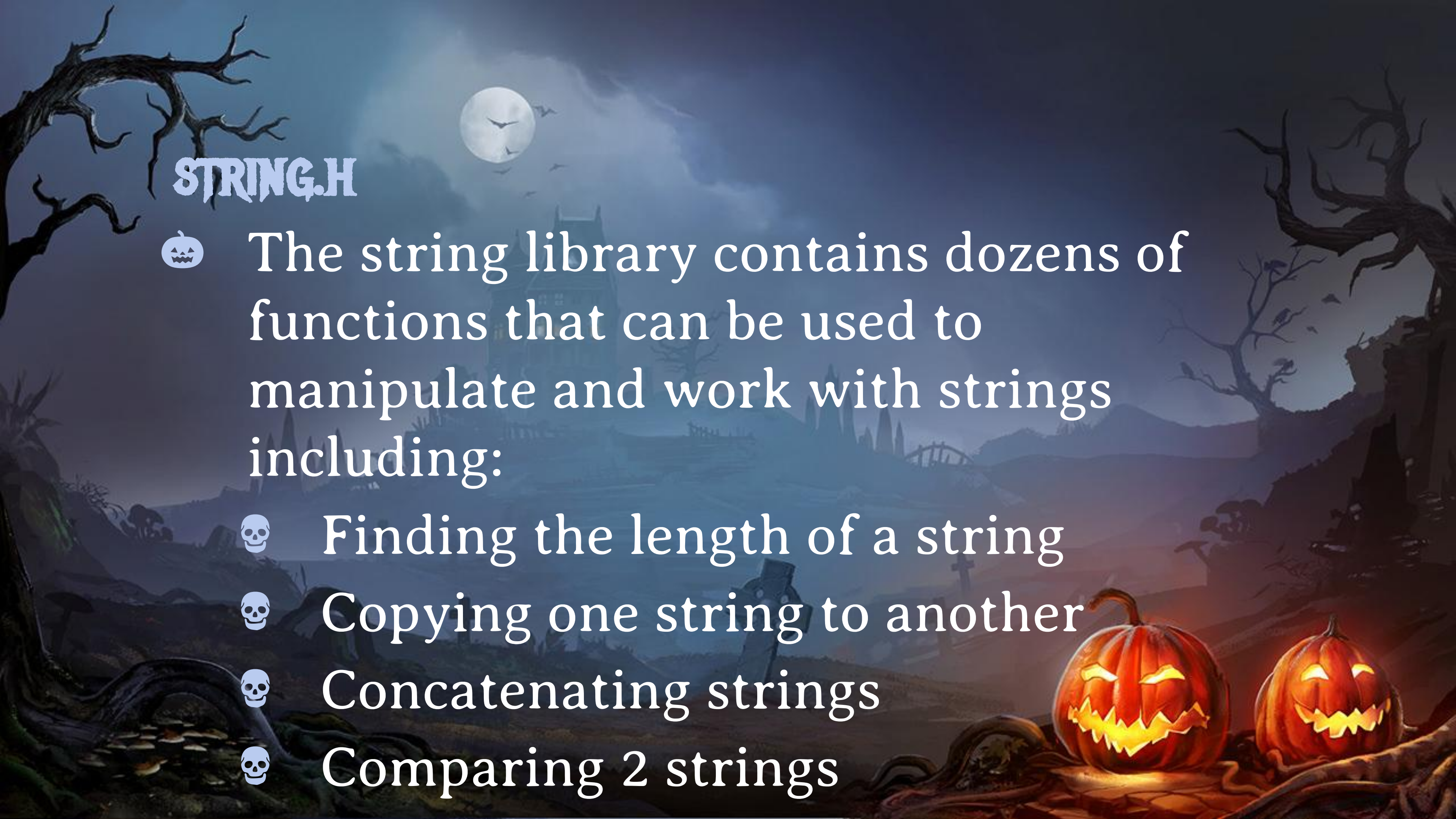
```c
#include <stdio.h>
#include <string.h>

int main() {
    char str[60];
    int length;

    //Input
    fgets(str, sizeof(str), stdin);

    //Remove the inputted newline
    length = strlen(str);
    if (str[length - 1] == '\n') {
        str[length - 1] = '\0';
    }

    //Output
    printf("%s\n", str);

    return 0;
}
```

Format Specifier for Strings

Since these character arrays are strings, we no longer have to print the character array one element at a time

We can use the format specifier %s in our print statements

ex. printf(“Our string %s”,hello);



STRING.H

- The string library contains dozens of functions that can be used to manipulate and work with strings including:
  - Finding the length of a string
  - Copying one string to another
  - Concatenating strings
  - Comparing 2 strings

Functions in the Library

- This library includes
  - char *strcat(char *restrict, const char *restrict);
  - int strcmp(const char *, const char *);
  - char *strcpy(char *restrict, const char *restrict);
  - size_t strlen(const char *);
  - char *strtok(char *restrict, const char *restrict);

We'll be using strlen today.

Feel free to experiment using other string functions

STRLEN

- The function strlen returns the number of characters in the string not including the null terminator.
- The format of strlen is:

  size_t strlen(const char *str)

- str – The string whose length is to be found
- size_t – Is an unsigned integer type meaning it cannot represent negative values.

On success the function returns the length of the string.

STRCPY

- The function strcpy will allow you to copy the contents of one string into another.
- The format of strcpy is:

  char * strcpy(char * dest, const char * src)

- dest – The pointer t the destination array where the content is to be copied.
- src – The string to be copied.

On success returns a pointer to the destination string dest.

STRCAT

- The function strcat is used to concatenate strings together. That is, it is used to insert the contents of one string to the end of the contents of another string. Also overwriting the null terminator ‘\0’ of the first string.
- The format of strcpy is:

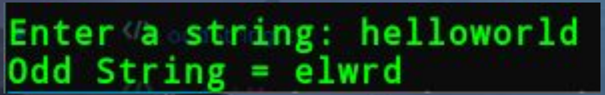
  char *strcat(char *dest, const char *src)
- dest – The pointer to the destination array, which should contain a C string, and should be large enough to contain the concatenated resulting string
- src – The string to be appended. This should not overlap the destination.
- On success, the function returns a pointer to the resulting string dest

Real Lab Exercise: Odd String

Write a program that prompts the user to enter a value :

1. a string that can hold a maximum of 20 characters.

It then creates a new string (that can hold a maximum of 10 characters), that contains all the odd-numbered indexed characters in originalS (odd not including index 0)



```
Enter a string: helloworld
Odd String = elwrd
```

oddString.c

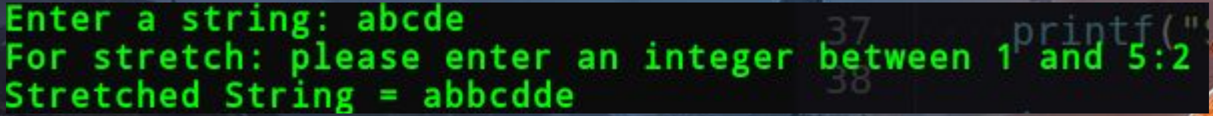
Lab Exercise: Stretch Me

Write a program that prompts the user to enter two values :

1. a number between 1 and 5
2. a string that can hold a maximum of 10 characters.

Check if number is within range

It then creates a new string (that can hold a maximum of 30 characters), that contains all characters in originalS - but the ones at odd-numbered indices are repeated numberToStretch times. (odd not including index 0)



```
Enter a string: abcde
For stretch: please enter an integer between 1 and 5:2
Stretched String = abbcdde
```

stretchme.c

Homework Problems: Due tomorrow @ 11:55 pm

- Each person must write their own solution, but the group should collaborate to identify the subproblems and then collaborate to identify possible solutions to each sub problem.
- Members of the group may leave when the entire group understands the homework task.
- Submit a single .c file
- MUST COMPILE OR IT IS A 0