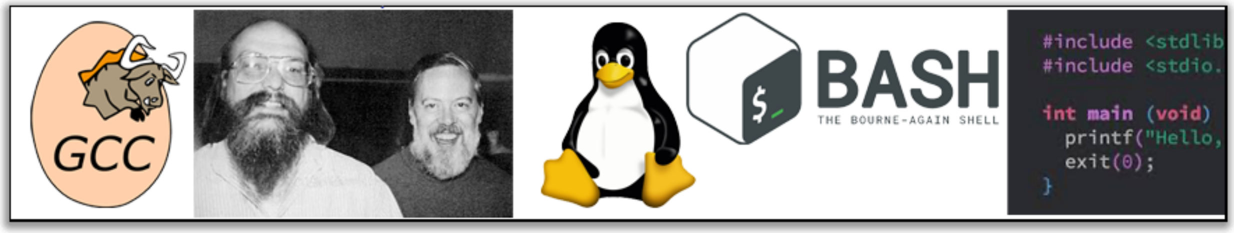

```
#include <stdlib
#include <stdio.

int main (void)
  printf("Hello,
  exit(0);
}
```

Assignment Day	Tuesday, Mar 16, 2021 may still make clarification modification after assignment date/time.
Due Date	**See schedule**

*** EDIT (03/25): You do not need to implement 'door functionality'

Learning Objectives:

- Deepened C Programmer knowledge
- Deepened UNIX knowledge
- UNIX Directory Tree Structure practice
- Recursion
- Demonstrate UNIX/C programming

Background Material:

Lab 7, tutorial 6 and tutorial 7- should provide nice sample programs.
Chapter 4

Program Description:

You will implement a simplified version of the UNIX command **find**, here called: **sfind** that recursively searches directories for files or directories that match a substring given on the command line.

The syntax of your command should look like this:

```
sfind <directory-where to start looking> -n substring-criteria -t <filetype>
```

As background what you (really) are implementing is the UNIX equivalent of:

```
find <directory-where to start looking> -name substring -print -type <filetype>
```

Here are some example uses:

- $ sfind ~maria -n exam-solutions.txt

  The above command line should find all the files and directories containing the string "exam-solutions.txt" starting from the directory ~maria OR in any of the subdirectories that descends from the ~maria. The files returned can be of any type since it is not defined by the -t flat.

- $ sfind . -n .txt -t f

  The above command line should find all the files and directories containing the string ".txt" in the current directory OR in any of the subdirectories that descends from the current directory. The files returned must be 'regular files' (the file types shold be at the same syntax as specified by the find command and to query the type you should use **lstat()**.

- $ sfind . -n maria

  The above command line is similar to the first example, except that it starts searching from the current directory, and searches for the substring 'maria'

If you prefer you can use getopt(), in that case for the current directory precede it with -s flag, e.g., sfind **-s** . -n maria

For the project you will use, `lstat()` instead of `stat()`. You used `lstat`() during the exam so it should be familiar, but just incase here is the man page of `lstat() / stat()` is (here).

Hints and Programming Strategies:

1. Read the manual page on the `find` command. Try it out in your own directory, e.g.,:
   ```
   $ find . -name Makefile -print
   ```
2. You need to use `lstat()` to determine the type of a file. Review the online documentation, and man page, as it is an essential part of your program.
3. Consider writing a function called `searchdir( char *dirname, char *findme , char type )`. That opens a directory called `dirname`, then read it entry by entry, and prints out names of items that match all search criteria. If `` `dirname´ `` is not null, then the function will only list files that match that pattern given in the parameter `findme`. If \`type´ is not '\0', then the function should (only) include items that are of the type specified.
4. In addition, if any entry in the directory is a directory, then `searchdir`() will need to create a new string that holds the new directory name and pass that string to itself. Use malloc(), or calloc() to create the string; a fixed size buffer is liable to run out when you least expect it.

Other Requirements:

- Each time a match is found, print the absolute path name for the matching file or directory.
- Test access permission before trying to change to a directory.
- Do not follow symbolic links.
- Your program must compile on nike, and be compiled with a single 'make' command.
- The programs should be robust and include appropriate error checks.
- You should use try to make your program as efficient as possible.
- Your program should be readable.
- You should comment your code.

You should maximize using system proggrams, i.e., use read(), write(), not use the 'f' commands expect for debugging purposed.

Submission:

You must submit the following files (i.e., all the files necessary to compile your program):
README.txt (how are "words" defined).
*.c (all your .c files, you should have at least 2 .c files)
*.h (all your .h files, you should have at least 1 .h file).
Makefile - must be run by a single:
make
it needs to use gcc -Wall & create the **sfind** executable.

video demo is required (2 days after due date of software)

survey monkey (tbd)

You will also need to make submit demo video (more on this later).

```
submit P3 csci-1730
```

NOTE 1: you need to be LOGGED ONTO odin.cs.uga when you execute the command