```
// Jessica Elkins
// CS332
// Project 2
// 3/3/30
// This program traverses a file hierchy and displays specific files based
// on the given command-line options

// TO COMPILE: gcc search.c -o search
// TO RUN: ./search <command-line options> <directoryname>

#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <unistd.h>
#include <string.h>

//global variables
int case1;
int fileSize;
char *substring;

typedef void MYFUNC(char *name, int length);

// when just ./search is executed
void fileTraversal(char *name, int length) {

        //to store DIR pointer returned from opendir
        DIR *dir;

        //to store pointer to structure return from readdir
        struct dirent *dirent;

        //opening directory
        dir = opendir(name);

        //if not able to open directory
        if(dir == NULL) {
                //print error message and terminate program
                printf("Error while opening directory. Exiting.\n");
                exit(-1);
        }

        //readdir returns NULL at end of directory or error
        while((dirent = readdir(dir)) != NULL) {
                //if path name is a directory
                if((dirent->d_type == DT_DIR) && (strcmp(dirent->d_name, ".") != 0) &&
(strcmp(dirent->d_name, "..") != 0)) {
                        //allocating size for path name
                        char pathName[BUFSIZ];

                        //using snprintf to format pathway name and storing it in pathN
ame
                        snprintf(pathName, sizeof(pathName), "%s/%s", name, dirent->d_n
ame);
                        printf(" \n");

                        //displaying directory name
                        printf("%*s %s \n", length, "", dirent->d_name);

                        //recursively call function to traverse directory
                        fileTraversal(pathName, length + 4);
                } else {
                        //if not directory, just list file name
```

```
                        printf("%*s %s \n", length, "", dirent->d_name);
                }
        }

        printf(" \n");

        //close directory
        closedir(dir);

}

// when ./search -S is executed
void includeFileSize(char *name, int length){
        DIR *dir;
        struct dirent *dirent;

        dir = opendir(name);

        if(dir == NULL){
                printf("Error while opening directory. Exiting. \n");
                exit(-1);
        }

        while((dirent = readdir(dir)) != NULL) {
                if((dirent->d_type == DT_DIR) && (strcmp(dirent->d_name, ".") != 0) &&
(strcmp(dirent->d_name, "..") != 0)){
                        char pathName[BUFSIZ];
                        snprintf(pathName, sizeof(pathName), "%s/%s", name, dirent->d_n
ame);
                        printf("\n");
                        printf("%*s %s [%d] \n", length, "", dirent->d_name, dirent->d_
reclen);
                        includeFileSize(pathName, length + 4);
                }else {
                        printf("%*s %s [%d] \n", length, "", dirent->d_name, dirent->d_
reclen);
                }
        }

        printf("\n");
        closedir(dir);
}

// when ./search -s <filesize> is executed
void fileSizeSearch(char *name, int length){
        DIR *dir;
        struct dirent *dirent;
        dir = opendir(name);

        if(dir == NULL){
                printf("Error while opening directory. Exiting\n");
                exit(-1);
        }

        while((dirent = readdir(dir)) != NULL){
                if((dirent->d_type == DT_DIR) && (strcmp(dirent->d_name, ".") != 0) &&
(strcmp(dirent->d_name, "..") != 0)){
                        char pathName[BUFSIZ];
                        snprintf(pathName, sizeof(pathName), "%s/%s", name, dirent->d_n
ame);
                        printf("\n");
                        if(dirent->d_reclen >= fileSize){
                                printf("%*s %s \n", length, "", dirent->d_name);
```

```
                        }
                        fileSizeSearch(pathName, length + 4);
                }else {
                        if(dirent->d_reclen >= fileSize){
                                printf("%*s %s \n", length, "", dirent->d_name);
                        }
                }
        }

        printf("\n");
        closedir(dir);
}

// when ./search -S -s <filesize> is executed
void sizeAndSizeSearch(char *name, int length){
        DIR *dir;
        struct dirent *dirent;
        dir = opendir(name);

        if(dir == NULL){
                printf("Error while opening directory. Exiting. \n");
                exit(-1);
        }

        while((dirent = readdir(dir)) != NULL){
                if((dirent->d_type == DT_DIR) && (strcmp(dirent->d_name, ".") != 0) &&
(strcmp(dirent->d_name, "..") != 0)){
                        char pathName[BUFSIZ];
                        snprintf(pathName, sizeof(pathName), "%s/%s", name, dirent->d_n
ame);
                        printf("\n");
                        if(dirent->d_reclen >= fileSize){
                                printf("%*s %s [%d] \n", length, "", dirent->d_name, di
rent->d_reclen);
                        }
                        sizeAndSizeSearch(pathName, length + 4);
                }else{
                        if(dirent->d_reclen >= fileSize){
                                printf("%*s %s [%d] \n", length, "", dirent->d_name, di
rent->d_reclen);
                        }
                }
        }

        printf("\n");
        closedir(dir);
}

// when ./search -f <substring> is executed
void substringSearch(char *name, int length){
        DIR *dir;
        struct dirent *dirent;
        dir = opendir(name);
        char *s;

        if(dir == NULL){
                printf("Error while opening directory. Exiting. \n");
                exit(-1);
        }

        while((dirent = readdir(dir)) != NULL){
                if((dirent->d_type == DT_DIR) && (strcmp(dirent->d_name, ".") != 0) &&
(strcmp(dirent->d_name, "..") != 0)){
```

```
                        char pathName[BUFSIZ];
                        snprintf(pathName, sizeof(pathName), "%s/%s", name, dirent->d_n
ame);
                        printf("\n");
                        if((s = strstr(dirent->d_name, substring)) != NULL){
                                printf("%*s %s \n", length, "", dirent->d_name);
                        }
                        substringSearch(pathName, length + 4);
                }else{
                        if((s = strstr(dirent->d_name, substring)) != NULL){
                                printf("%*s %s \n", length, "", dirent->d_name);
                        }
                }
        }

        printf("\n");
        closedir(dir);
}

// when ./search -S -f <substring> is executed
void sizeAndSubstring(char *name, int length){
        DIR *dir;
        struct dirent *dirent;
        dir = opendir(name);
        char *s;

        if(dir == NULL){
                printf("Error opening the directory. Exiting. \n");
                exit(-1);
        }

        while((dirent = readdir(dir)) != NULL){
                if((dirent->d_type == DT_DIR) && (strcmp(dirent->d_name, ".") != 0) &&
(strcmp(dirent->d_name, "..") != 0)){
                        char pathName[BUFSIZ];
                        snprintf(pathName, sizeof(pathName), "%s/%s", name, dirent->d_n
ame);
                        printf("\n");
                        if((s = strstr(dirent->d_name, substring)) != NULL){
                                printf("%*s %s [%d] \n", length, "", dirent->d_name, di
rent->d_reclen);
                        }
                        sizeAndSubstring(pathName, length + 4);
                }else{
                        if((s = strstr(dirent->d_name, substring)) != NULL){
                                printf("%*s %s [%d] \n", length, "", dirent->d_name, di
rent->d_reclen);
                        }
                }
        }

        printf("\n");
        closedir(dir);
}

// when ./search -s <filesize> -f <substring> is executed
void sizeSearchAndSubstring(char *name, int length){
        DIR *dir;
        struct dirent *dirent;
        dir = opendir(name);
        char *s;

        if(dir == NULL){
```

```
                printf("Error opening the directory. Exiting. \n");
                exit(-1);
        }

        while((dirent = readdir(dir)) != NULL){
                if((dirent->d_type == DT_DIR) && (strcmp(dirent->d_name, ".") != 0) &&
(strcmp(dirent->d_name, "..") != 0)){
                        char pathName[BUFSIZ];
                        snprintf(pathName, sizeof(pathName), "%s/%s", name, dirent->d_n
ame);
                        printf("\n");
                        if(((s = strstr(dirent->d_name, substring)) != NULL) && (dirent
->d_reclen >= fileSize)){
                                printf("%*s %s \n", length, "", dirent->d_name);
                        }
                        sizeSearchAndSubstring(pathName, length + 4);
                }else{
                        if(((s = strstr(dirent->d_name, substring)) != NULL) && (dirent
->d_reclen >= fileSize)){
                                printf("%*s %s \n", length, "", dirent->d_name);
                        }
                }
        }

        printf("\n");
        closedir(dir);
}

// when ./search -S -s <filesize> -f <substring> is executed
void allOfThem(char *name, int length){
        DIR *dir;
        struct dirent *dirent;
        dir = opendir(name);
        char *s;

        if(dir == NULL){
                printf("Error opening directory. Exiting. \n");
                exit(-1);
        }

        while((dirent = readdir(dir)) != NULL){
                if((dirent->d_type == DT_DIR) && (strcmp(dirent->d_name, ".") != 0) &&
(strcmp(dirent->d_name, "..") != 0)){
                        char pathName[BUFSIZ];
                        snprintf(pathName, sizeof(pathName), "%s/%s", name, dirent->d_n
ame);
                        printf("\n");
                        if(((s = strstr(dirent->d_name, substring)) != NULL) && (dirent
->d_reclen >= fileSize)){
                                printf("%*s %s [%d] \n", length, "", dirent->d_name, di
rent->d_reclen);
                        }
                        allOfThem(pathName, length + 4);
                }else {
                        if(((s = strstr(dirent->d_name, substring)) != NULL) && (dirent
->d_reclen >= fileSize)){
                                printf("%*s %s [%d] \n", length, "", dirent->d_name, di
rent->d_reclen);
                        }
                }
        }

        printf("\n");
```

```
        closedir(dir);
}

// function pointer
void opfunc(char *name, int length, MYFUNC *f) {
        //calls the function name given as an argument
        f(name, length);
}

int main(int argc, char **argv) {
        //if only one command line argument was given
        if(argc < 2) {
                argv[1] = ".";
        }

        // to store getopt return
        int g = 0;

        // -S
        case1 = 0;
        // -s fileSize
        fileSize = 0;
        // -f substring
        substring = NULL;
        // pathway to pass to traversal function
        char *name = NULL;

        while((g = getopt(argc, argv, "Ss:f:")) != -1) {
                switch(g) {
                        case 'S':  // if -S was entered
                          case1++; // add one to case!
                          break;
                        case 's':  // if -s fileSize was entered
                          fileSize = atoi(optarg);  //store file size
                          break;
                        case 'f':  // if -f substring was entered
                          substring = optarg;  // store the substring
                          break;
                }
        }

        //finding the given pathway at the end of the arguments listed
        int index = optind;
        // if no pathway was given, set pathway to "."
        if(argv[index] == NULL) {
                argv[index] = ".";
        }
        name = argv[index];
        printf("pathname: %s\n", name);

        if((case1 == 0) && (fileSize == 0) && (substring == NULL)){
                // ./search .
                opfunc(name, 0, fileTraversal);
        }else if((case1 == 1) && (fileSize == 0) && (substring == NULL)){
                // ./search -S
                opfunc(name, 0, includeFileSize);
        }else if((case1 == 0) && (fileSize != 0) && (substring == NULL)){
                // ./search -s 1024
                opfunc(name, 0, fileSizeSearch);
        }else if((case1 == 0) && (fileSize == 0) && (substring != NULL)){
                // ./search -f jpg
                opfunc(name, 0, substringSearch);
        }else if((case1 == 1) && (fileSize != 0) && (substring == NULL)){
```

```
                // ./search -S -s 1024
                opfunc(name, 0, sizeAndSizeSearch);
        }else if((case1 == 1) && (fileSize == 0) && (substring != NULL)){
                // ./search -S -f jpg
                opfunc(name, 0, sizeAndSubstring);
        }else if((case1 == 0) && (fileSize != 0) && (substring != NULL)){
                // ./search -s -f
                opfunc(name, 0, sizeSearchAndSubstring);
        }else if((case1 == 1) && (fileSize != 0) && (substring != NULL)){
                // ./search -S -s -f
                opfunc(name, 0, allOfThem);
        }

        return 0;

}
```