```
// Name: Jessica Elkins
// BlazerID: jelkins3
// Assignment: Project 3 for CS332 Spring 2020
// Date: 4/2/20
// Description: 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>
#include <sys/stat.h>
#include <sys/wait.h>

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

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;
        struct stat statbuf;

        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");
                        stat(dirent->d_name, &statbuf);
                        printf("%*s %s [%d bytes] \n", length, "", dirent->d_name, stat
buf.st_size);
                        includeFileSize(pathName, length + 4);
                        printf("\n");
                }else {
                        stat(dirent->d_name, &statbuf);
                        if( (strcmp(dirent->d_name, ".") != 0) && (strcmp(dirent->d_nam
e, "..") != 0) ) {
                                printf("%*s %s [%d bytes] \n", length, "", dirent->d_na
me, statbuf.st_size);
                                printf("\n");
                        } else {
                                printf("%*s %s \n", length, "", dirent->d_name);
                                printf("\n");
                        }
                }
        }

        closedir(dir);
}

// when ./search -s <filesize> is executed
void fileSizeSearch(char *name, int length){
        DIR *dir;
        struct dirent *dirent;
        struct stat statbuf;
        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);
                        stat(dirent->d_name, &statbuf);
                        if(statbuf.st_size >= fileSize){
                                printf("\n");
                                printf("%*s %s \n", length, "", dirent->d_name);
                                printf("\n");
                        }
                        fileSizeSearch(pathName, length + 4);
                }else {
                        stat(dirent->d_name, &statbuf);
                        if(statbuf.st_size >= 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;
        struct stat statbuf;
        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);
                        stat(dirent->d_name, &statbuf);
                        if(statbuf.st_size >= fileSize){
                                printf("\n");
                                printf("%*s %s [%d] \n", length, "", dirent->d_name, st
atbuf.st_size);
                                printf("\n");
                        }
                        sizeAndSizeSearch(pathName, length + 4);
                }else{
                        stat(dirent->d_name, &statbuf);
                        if(statbuf.st_size >= fileSize){
                                printf("%*s %s [%d] \n", length, "", dirent->d_name, st
atbuf.st_size);
                                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);

                        if((s = strstr(dirent->d_name, substring)) != NULL){
                                printf("\n");
                                printf("%*s %s \n", length, "", dirent->d_name);
                                printf("\n");
                        }
                        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;
        struct stat statbuf;
        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);
                        stat(dirent->d_name, &statbuf);
                        if((s = strstr(dirent->d_name, substring)) != NULL){
                                printf("\n");
                                printf("%*s %s [%d bytes] \n", length, "", dirent->d_na
me, statbuf.st_size);
                                printf("\n");
                        }
                        sizeAndSubstring(pathName, length + 4);
                }else{
                        stat(dirent->d_name, &statbuf);
                        if((s = strstr(dirent->d_name, substring)) != NULL){
                                printf("%*s %s [%d bytes] \n", length, "", dirent->d_na
me, statbuf.st_size);
                                printf("\n");
                        }
                }
        }

        closedir(dir);
}

// when ./search -s <filesize> -f <substring> is executed
void sizeSearchAndSubstring(char *name, int length){
        DIR *dir;
        struct dirent *dirent;
        struct stat statbuf;
        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);
                        stat(dirent->d_name, &statbuf);
                        if(((s = strstr(dirent->d_name, substring)) != NULL) && (statbu
f.st_size >= fileSize)){
                                printf("\n");
                                printf("%*s %s \n", length, "", dirent->d_name);
                                printf("\n");
                        }
                        sizeSearchAndSubstring(pathName, length + 4);
                }else{
                        stat(dirent->d_name, &statbuf);
                        if(((s = strstr(dirent->d_name, substring)) != NULL) && (statbu
f.st_size >= 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;
        struct stat statbuf;
        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);
                        stat(dirent->d_name, &statbuf);
                        if(((s = strstr(dirent->d_name, substring)) != NULL) && (statbu
f.st_size >= fileSize)){
                                printf("\n");
                                printf("%*s %s [%d bytes] \n", length, "", dirent->d_na
me, statbuf.st_size);
                                printf("\n");
                        }
                        allOfThem(pathName, length + 4);
                }else {
                        stat(dirent->d_name, &statbuf);
                        if(((s = strstr(dirent->d_name, substring)) != NULL) && (statbu
f.st_size >= fileSize)){
                                printf("\n");
                                printf("%*s %s [%d bytes] \n", length, "", dirent->d_na
me, statbuf.st_size);
                                printf("\n");
                        }
                }
        }

        closedir(dir);
}

void forkFunc(char *command, char **args){
        int i, count, length;
        length = strlen(command);

        for(i = 0, args[0] = &command[0], count = 1; i < length; i++){
                if(command[i] == ' '){
                        command[i] = '\0';
                        args[count++] = &command[i+1];
                }
        }

        args[count] = (char *)NULL;

        int status;
        pid_t pid;
        pid = fork();

        if(pid == 0){
                execvp(args[0], args);
                perror("exec");
                exit(-1);
        }else if (pid > 0) {
                wait(&status);
                if(WIFEXITED(status)){
                        //child exited normal
                }else{
                        printf("Child process did not terminate normally! \n");
                }
        }
}

// when ./search -e [command] is executed
void execute(char *name, int length){
        DIR *dir;
        struct dirent *dirent;
        dir = opendir(name);
        char command2[BUFSIZ];
        char *args[BUFSIZ];


        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");
                        printf("%*s %s \n", length, "", dirent->d_name);
                        strcpy(command2, command);
                        strcat(command2, " ");
                        strcat(command2, pathName);
                        forkFunc(command2, args);
                        printf("\n");
                        execute(pathName, length + 4);
                }else{
                        printf("\n");
                        printf("%*s %s \n", length, "", dirent->d_name);
                        char pathName[BUFSIZ];
                        snprintf(pathName, sizeof(pathName), "%s/%s", name, dirent->d_n
ame);
                        strcpy(command2, command);
                        strcat(command2, " ");
                        strcat(command2, pathName);
                        forkFunc(command2, args);
                }
        }

        closedir(dir);
}

// when ./search -S -e [command] is executed
void executeAndSize(char *name, int length){
        DIR *dir;
        struct dirent *dirent;
        struct stat statbuf;
        char *s;
        char command2[BUFSIZ];
        char *args[BUFSIZ];

        dir = opendir(name);
        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");
                        stat(dirent->d_name, &statbuf);
                        printf("%*s %s [%d bytes] \n", length, "", dirent->d_name, stat
buf.st_size);
                        strcpy(command2, command);
                        strcat(command2, " ");
                        strcat(command2, pathName);
                        forkFunc(command2, args);
                        printf("\n");
                        executeAndSize(pathName, length + 4);
                } else {
                        printf("\n");
                        stat(dirent->d_name, &statbuf);
                        printf("%*s %s [%d bytes] \n", length, "", dirent->d_name, stat
buf.st_size);
                        char pathName[BUFSIZ];
                        snprintf(pathName, sizeof(pathName), "%s/%s", name, dirent->d_n
ame);
                        strcpy(command2, command);
                        strcat(command2, " ");
                        strcat(command2, pathName);
                        forkFunc(command2, args);

                }
        }

        closedir(dir);
}

// when ./search -s [fileSize] -e [command] is executed
void executeAndSizeSearch(char *name, int length){
        DIR *dir;
        struct dirent *dirent;
        char command2[BUFSIZ];
        char *args[BUFSIZ];
        struct stat statbuf;

        dir = opendir(name);
        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);
                        stat(dirent->d_name, &statbuf);
                        if(statbuf.st_size >= fileSize){
                                printf("\n");
                                printf("%*s %s \n", length, "", dirent->d_name);
                                strcpy(command2, command);
                                strcat(command2, " ");
                                strcat(command2, pathName);
                                forkFunc(command2, args);
                                printf("\n");
                                executeAndSizeSearch(pathName, length + 4);
                        }
                }else{
                        stat(dirent->d_name, &statbuf);
                        if(statbuf.st_size >= fileSize){
                                char pathName[BUFSIZ];
                                snprintf(pathName, sizeof(pathName), "%s/%s", name, dir
ent->d_name);
                                printf("\n");
                                printf("%*s %s \n", length, "", dirent->d_name);
                                strcpy(command2, command);
                                strcat(command2, " ");
                                strcat(command2, pathName);
                                forkFunc(command2, args);
                                printf("\n");
                        }

                }
        }
        closedir(dir);
}

//when ./search -f [substring] -e [command] is executed
void executeAndSubstring(char *name, int length){
        DIR *dir;
        struct dirent *dirent;
        char command2[BUFSIZ];
        char *args[BUFSIZ];
        char *s;

        dir = opendir(name);
        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);
                        if((s = strstr(dirent->d_name, substring)) != NULL){
                                printf("\n");
                                printf("%*s %s \n", length, "", dirent->d_name);
                                strcpy(command2, command);
                                strcat(command2, " ");
                                strcat(command2, pathName);
                                forkFunc(command2, args);
                                printf("\n");
                                executeAndSubstring(pathName, length + 4);
                        }
                }else{
                        if((s = strstr(dirent->d_name, substring)) != NULL){
                                char pathName[BUFSIZ];
                                snprintf(pathName, sizeof(pathName), "%s/%s", name, dir
ent->d_name);
                                printf("\n");
                                printf("%*s %s \n", length, "", dirent->d_name);
                                strcpy(command2, command);
                                strcat(command2, " ");
                                strcat(command2, pathName);
                                forkFunc(command2, args);
                                printf("\n");
                        }

                }
        }
        closedir(dir);
}

// when ./search -S -s [fileSize] -e [command]
void exeAndSizeAndSizeSearch(char *name, int length){
        DIR *dir;
        struct dirent *dirent;
        struct stat statbuf;
        char command2[BUFSIZ];
        char *args[BUFSIZ];

        dir = opendir(name);
        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);
                        stat(dirent->d_name, &statbuf);
                        if(statbuf.st_size >= fileSize){
                                printf("\n");
                                printf("%*s %s [%d bytes] \n", length, "", dirent->d_na
me, statbuf.st_size);
                                strcpy(command2, command);
                                strcat(command2, " ");
                                strcat(command2, pathName);
                                forkFunc(command2, args);
                                printf("\n");
                                exeAndSizeAndSizeSearch(pathName, length + 4);
                        }
                }else{
                        stat(dirent->d_name, &statbuf);
                        if(statbuf.st_size >= fileSize){
                                printf("\n");
                                printf("%*s %s [%d bytes] \n", length, "", dirent->d_na
me, statbuf.st_size);
                                char pathName[BUFSIZ];
                                snprintf(pathName, sizeof(pathName), "%s/%s", name, dir
ent->d_name);
                                strcpy(command2, command);
                                strcat(command2, " ");
                                strcat(command2, pathName);
                                forkFunc(command2, args);
                                printf("\n");
                        }

                }
        }
        closedir(dir);
}

// when ./search -S -f [substring] -e [command] is executed
void S_f_e(char *name, int length){
        DIR *dir;
        struct dirent *dirent;
        struct stat statbuf;
        char *s;
        char command2[BUFSIZ];
        char *args[BUFSIZ];

        dir = opendir(name);
        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);
                        stat(dirent->d_name, &statbuf);
                        if((s = strstr(dirent->d_name, substring)) != NULL){
                                printf("\n");
                                printf("%*s %s [%d bytes] \n", length, "", dirent->d_na
me, statbuf.st_size);
                                strcpy(command2, command);
                                strcat(command2, " ");
                                strcat(command2, pathName);
                                forkFunc(command2, args);
                                printf("\n");
                                S_f_e(pathName, length + 4);
                        }
                }else{
                        if((s = strstr(dirent->d_name, substring)) != NULL){
                                stat(dirent->d_name, &statbuf);
                                printf("\n");
                                printf("%*s %s [%d bytes] \n", length, "", dirent->d_na
me, statbuf.st_size);
                                char pathName[BUFSIZ];
                                snprintf(pathName, sizeof(pathName), "%s/%s", name, dir
ent->d_name);
                                strcpy(command2, command);
                                strcat(command2, " ");
                                strcat(command2, pathName);
                                forkFunc(command2, args);
                                printf("\n");
                        }
                }
        }

        closedir(dir);
}

// when ./search -s [fileSize] -f [substring] -e [command]
void size_f_e(char *name, int length){
        DIR *dir;
        struct dirent *dirent;
        struct stat statbuf;
        char *s;
        char command2[BUFSIZ];
        char *args[BUFSIZ];

        dir = opendir(name);
        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);
                        stat(dirent->d_name, &statbuf);
                        if( ((s = strstr(dirent->d_name, substring)) != NULL) && (statb
uf.st_size >= fileSize)){
                                printf("\n");
                                printf("%*s %s \n", length, "", dirent->d_name);
                                strcpy(command2, command);
                                strcat(command2, " ");
                                strcat(command2, pathName);
                                forkFunc(command2, args);
                                printf("\n");
                                size_f_e(pathName, length + 4);
                        }
                }else{
                        stat(dirent->d_name, &statbuf);
                        if( ((s = strstr(dirent->d_name, substring)) != NULL) && (statb
uf.st_size >= fileSize)){
                                printf("\n");
                                printf("%*s %s \n", length, "", dirent->d_name);
                                char pathName[BUFSIZ];
                                snprintf(pathName, sizeof(pathName), "%s/%s", name, dir
ent->d_name);
                                strcpy(command2, command);
                                strcat(command2, " ");
                                strcat(command2, pathName);
                                forkFunc(command2, args);
                                printf("\n");
                        }
                }
        }

        closedir(dir);
}

// when ./search -S -s [fileSize] -f [substring] -e [command]
void everything(char *name, int length){
        DIR *dir;
        struct dirent *dirent;
        struct stat statbuf;
        char *s;
        char command2[BUFSIZ];
        char *args[BUFSIZ];

        dir = opendir(name);
        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);
                        stat(dirent->d_name, &statbuf);
                        if( ((s = strstr(dirent->d_name, substring)) != NULL) && (statb
uf.st_size >= fileSize)){
                                printf("\n");
                                printf("%*s %s [%d bytes] \n", length, "", dirent->d_na
me, statbuf.st_size);
                                strcpy(command2, command);
                                strcat(command2, " ");
                                strcat(command2, pathName);
                                forkFunc(command2, args);
                                printf("\n");
                                everything(pathName, length + 4);
                        }
                }else{
                        stat(dirent->d_name, &statbuf);
                        if( ((s = strstr(dirent->d_name, substring)) != NULL) && (statb
uf.st_size >= fileSize)){
                                printf("\n");
                                printf("%*s %s [%d bytes] \n", length, "", dirent->d_na
me, statbuf.st_size);
                                char pathName[BUFSIZ];
                                snprintf(pathName, sizeof(pathName), "%s/%s", name, dir
ent->d_name);
                                strcpy(command2, command);
                                strcat(command2, " ");
                                strcat(command2, pathName);
                                forkFunc(command2, args);
                                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;
        // -e command
        command = NULL;
        // pathway to pass to traversal function
        char *name = NULL;

        while((g = getopt(argc, argv, "Ss:f:e:")) != -1) {
                switch(g) {
                        case 'S':  // if -S was entered
                          case1++; // add one to case1
                          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;
                        case 'e':  // if -e [command] was entered
                          command = optarg;  // store the command
                          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);
        printf("\n");

        if((case1 == 0) && (fileSize == 0) && (substring == NULL) && (command == NULL))
{
                // ./search .
                opfunc(name, 0, fileTraversal);
        }else if((case1 == 1) && (fileSize == 0) && (substring == NULL) && (command ==
NULL)){
                // ./search -S
                opfunc(name, 0, includeFileSize);
        }else if((case1 == 0) && (fileSize != 0) && (substring == NULL) && (command ==
NULL)){
                // ./search -s 1024
                opfunc(name, 0, fileSizeSearch);
        }else if((case1 == 0) && (fileSize == 0) && (substring != NULL) && (command ==
NULL)){
                // ./search -f jpg
                opfunc(name, 0, substringSearch);
        }else if((case1 == 1) && (fileSize != 0) && (substring == NULL) && (command ==
NULL)){
                // ./search -S -s 1024
                opfunc(name, 0, sizeAndSizeSearch);
        }else if((case1 == 1) && (fileSize == 0) && (substring != NULL) && (command ==
NULL)){
                // ./search -S -f jpg
                opfunc(name, 0, sizeAndSubstring);
        }else if((case1 == 0) && (fileSize != 0) && (substring != NULL) && (command ==
NULL)){
                // ./search -s -f
                opfunc(name, 0, sizeSearchAndSubstring);
        }else if((case1 == 1) && (fileSize != 0) && (substring != NULL) && (command ==
NULL)){
                // ./search -S -s -f
                opfunc(name, 0, allOfThem);
        }else if((case1 == 0) && (fileSize == 0) && (substring == NULL) && (command !=
NULL)){
                // ./search -e [command]
                opfunc(name, 0, execute);
        }else if((case1 == 1) && (fileSize == 0) && (substring == NULL) && (command !=
NULL)){
                // ./search -S -e [command]
                opfunc(name, 0, executeAndSize);
        }else if((case1 == 0) && (fileSize != 0) && (substring == NULL) && (command !=
NULL)){
                // ./search -s [fileSize] -e [command]
                opfunc(name, 0, executeAndSizeSearch);
        }else if((case1 == 0) && (fileSize == 0) && (substring != NULL) && (command !=
NULL)){
                // ./search -f [substring] -e [command]
                opfunc(name, 0, executeAndSubstring);
        }else if((case1 == 1) && (fileSize != 0) && (substring == NULL) && (command !=
NULL)){
                // ./search -S -s [fileSize] -e [command]
                opfunc(name, 0, exeAndSizeAndSizeSearch);
        }else if((case1 == 1) && (fileSize == 0) && (substring != NULL) && (command !=
NULL)){
                // ./search -S -f [substring] -e [command]
                opfunc(name, 0, S_f_e);
        }else if((case1 == 0) && (fileSize != 0) && (substring != NULL) && (command !=
NULL)){
                // ./search -s [fileSize] -f [substring] -e [command]
                opfunc(name, 0, size_f_e);
        }else if((case1 == 1) && (fileSize != 0) && (substring != NULL) && (command !=
NULL)){
                // ./search -S -s [fileSize] -f [substring] -e [command]
                opfunc(name, 0, everything);
        }

        return 0;

}
```