```c
/*
Name: Sri Vastava Rangaraju Naga Venkata
BlazerId:  rsri21us
Project #: 2
To compile: gcc search.c
To run:  ./a.out <commands>
*/

#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include<string.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<unistd.h>
#include <errno.h>

typedef int MYFUNC(char *dir);

char *filetype(unsigned char type) {  // This code returns the file type (a String)
    char *str;
    switch(type) {
    case DT_BLK: str = "block device"; break;
    case DT_CHR: str = "character device"; break;
    case DT_DIR: str = "directory"; break;
    case DT_FIFO: str = "named pipe (FIFO)"; break;
    case DT_LNK: str = "symbolic link"; break;
    case DT_REG: str = "regular file"; break;
    case DT_SOCK: str = "UNIX domain socket"; break;
    case DT_UNKNOWN: str = "unknown file type"; break;
    default: str = "UNKNOWN";
    }
    return str;
  }

int getFileSize(char *filename){

    struct stat sb;

    if (stat(filename, &sb) == -1) {
        perror("stat");
        exit(EXIT_FAILURE);
    }

    return(sb.st_size);
}
void printFileHierarchy(char *path, int isS, int isF, int fileSize, char *substring)
{
    int count=1, flag=0;
    struct dirent *dirent;
    DIR *parentDir;
    parentDir = opendir (path);
    char *dir_path;

    if (parentDir == NULL) {
      printf("Cannot open the Directory");
      return;
    }

    while((dirent = readdir(parentDir)) != NULL)
    {
        char *tmp = malloc(sizeof(path)+sizeof("/")+sizeof(dirent->d_name));
        strcpy(tmp, path);
        strcat(tmp,"/");
        strcat(tmp, dirent->d_name);


      if(strcmp(dirent->d_name, ".") != 0 && strcmp(dirent->d_name, "..") != 0)
      {
        if( strcmp(filetype(dirent->d_type), "directory") == 0)
        {

              if(isS == 1)
              {
                  if(getFileSize(tmp)>=fileSize)
                   printf ("[%d] %s (%d)\n", count, dirent->d_name, getFileSize(tmp));
              }
              else if(isF ==1)
              {
                if(strstr(dirent->d_name, substring))
                    printf("[%d] %s \n", count, dirent->d_name);
              }
              else if(strstr(dirent->d_name, substring)==0 && getFileSize(tmp)>=fileSize)
              {
                    printf ("[%d] %s (%d)\n", count, dirent->d_name, getFileSize(tmp));
                    flag=1;
              }
              if(strcmp(substring, "dir")==0)
              {
                  printf ("[%d] %s (%d)\n", count, dirent->d_name, getFileSize(tmp));
              }
            else{
                    printf ("[%d] %s (%d)\n", count, dirent->d_name, getFileSize(tmp));
               }
                    dir_path = malloc(sizeof(path)+sizeof("/")+sizeof(dirent->d_name));
                    strcpy(dir_path, path);
                    strcat(dir_path,"/");
                    strcat(dir_path, dirent->d_name);
                    printFileHierarchy(dir_path, isS, isF, fileSize, substring);

        }
        else
        {
            if(isS == 1)
              {
                  if(getFileSize(tmp)>=fileSize)
                   printf ("[%d] %s (%d)\n", count, dirent->d_name, getFileSize(tmp));
              }
            else if(isF ==1)
            {
                if(strstr(dirent->d_name, substring))
                 printf("[%d] %s \n", count, dirent->d_name);
            }
            else if(strstr(dirent->d_name, substring) && getFileSize(tmp)>=fileSize)
              {
                    printf ("[%d] %s (size:%d)\n", count, dirent->d_name, getFileSize(tmp
));
              }
              if(strcmp(substring, "reg")==0)
              {
                  printf ("[%d] %s (%d)\n", count, dirent->d_name, getFileSize(tmp));
              }
              else{
                    printf ("[%d] %s (%d)\n", count, dirent->d_name, getFileSize(tmp));
            }
        }
        count++;
      }
    }
    closedir (parentDir);
}



void tempMethod(int argc, char **argv)
{
    int filesize=0;
    char *substring="";
    int isS, isF;
    if(argc == 7){
        if(strcmp(argv[2],"-s")==0 && strcmp(argv[4],"-f")==0)
        {
            isS = 0, isF = 0, filesize = atoi(argv[3]), substring = argv[5];
            printFileHierarchy(argv[6], isS, isF, filesize, substring);
        }
        else if(strcmp(argv[2],"-f")==0 && strcmp(argv[4],"-s")==0)
        {
            isS = 0, isF = 0, filesize = atoi(argv[3]), substring = argv[5];
            printFileHierarchy(argv[6], isS, isF, filesize, substring);
        }
    }

    if(argc == 6){
        if(strcmp(argv[2],"-s")==0 && strcmp(argv[4],"-f")==0)
        {
            isS = 0, isF = 0, filesize = atoi(argv[3]), substring = argv[5];
            printFileHierarchy(".", isS, isF, filesize, substring);
        }
        else if(strcmp(argv[2],"-f")==0 && strcmp(argv[4],"-s")==0)
        {
            isS = 0, isF = 0, filesize = atoi(argv[5]), substring = argv[3];
            printFileHierarchy(".", isS, isF, filesize, substring);
        }
    }
    if(argc ==5){
        if(strcmp(argv[1],"-s")==0 && strcmp(argv[3],"-f")==0)
        {
            isS = 0, isF = 0, filesize = atoi(argv[2]), substring = argv[4];
            printFileHierarchy(".", isS, isF, filesize, substring);
        }
        else if(strcmp(argv[1],"-f")==0 && strcmp(argv[3],"-s")==0)
        {
            isS = 0, isF = 0, filesize = atoi(argv[4]), substring = argv[2];
            printFileHierarchy(".", isS, isF, filesize, substring);
        }
    }
    if(argc == 4){
        if(strcmp(argv[1],"-s")==0)
        {
            isS = 1, isF = 0, filesize = atoi(argv[2]), substring = "";
            printFileHierarchy(argv[3], isS, isF, filesize, substring);
        }
        else if(strcmp(argv[1],"-f")==0)
        {
            isS = 0, isF = 1, filesize = 0, substring = argv[2];
            printFileHierarchy(argv[3], isS, isF, filesize, substring);
        }
    }
    if(argc == 3)
    {
        if(strcmp(argv[1],"-S")==0)
        {
            isS =0, isF=0, filesize=0, substring="";
            printFileHierarchy(argv[2], isS, isF, filesize, substring);
        }
        if(strcmp(argv[1],"-s")==0)
        {
            isS = 1, isF = 0, filesize = atoi(argv[2]), substring = "";
            printFileHierarchy(".", isS, isF, filesize, substring);
        }
        if(strcmp(argv[1], "-f")==0)
        {
             isS = 0, isF = 1, filesize = 0, substring = argv[2];
            printFileHierarchy(".", isS, isF, filesize, substring);
        }
        if(strcmp(argv[1], "-t")==0 && strcmp(argv[2],"d")==0)
        {
            filesize = 9, substring = "dir";
            printFileHierarchy(".", isS, isF, filesize, substring);
        }
        if(strcmp(argv[1], "-t")==0 && strcmp(argv[2],"f")==0)
        {
             filesize = 19, substring = "reg";
            printFileHierarchy(".", isS, isF, filesize, substring);
        }

    }

}


int main(int argc, char *argv[])
{
    int filesize=0;
    char *substring="";
    int isS=0, isF=0;

    if(argc == 1)
       printFileHierarchy(".", isS, isF, filesize, substring);
    else if(argc == 2)
    {
       printFileHierarchy(argv[1], isS, isF, filesize, substring);
    }
    else
    {
        int option;
        while(( option = getopt(argc, argv, "S:f:s:t:")) != -1 )
        {
            switch (option){
                case 'S':
                    tempMethod(argc, argv);
                    break;
                case 's':
                    tempMethod(argc, argv);
                    break;
                case 'f':
                    tempMethod(argc, argv);
                    break;
                case 't':
                    tempMethod(argc, argv);
                    break;
                case '?':
                    if(optopt == 'c')
                        fprintf(stderr, "Option -%c needs argument\n", optopt);
                    if(optopt == 'f')
                        fprintf(stderr, "Option -%c needs argument\n", optopt);
                    else
                        fprintf(stderr, "unknown option -%c. \n", optopt);
                    break;

                default:
                    fprintf(stderr, "getopt");
                    break;
                }
        }

    }
     return 0;
}
```