```
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include "dos2sd.h"

/*WAS MADE BY YUSUF ADISAPUTRO 213533088*/
static void extract(FILE *fp, struct ATRSSDISK *disk, char id[])
{
   printf("You have input %s \n\n", id);

        int sector, entry, i, count, start, baseFileNumber;
        char name[9], ext[4];

        int detectedStart, detectedCount;

        baseFileNumber = 0;
        for(sector=361;sector<=368;sector++) {
                for(entry=0;entry<ATR_SECTOR_SIZE;entry+=16) {
                        if(disk->sector[sector-1][entry] == 0x042) {
                                for(i=0;i<8;i++)
                                        name[i] = disk->sector[sector-1][entry+5+i];
                                name[8] = '\0';
                                for(i=0;i<3;i++)
                                        ext[i] = disk->sector[sector-1][entry+13+i];
                                ext[3] = '\0';
                                count = disk->sector[sector-1][entry+1]|disk->sector[se
ctor-1][entry+2]<<8;
                                start = disk->sector[sector-1][entry+3]|disk->sector[se
ctor-1][entry+4]<<8;

                                int result = checkingFile(name, id);
                                int result2 = checkingFileExt(ext, id);
                                if(result == 1 && result2 == 1){
                                        detectedStart = start;
                                        detectedCount = count;
                                        savingToLocal(stdout, disk, detectedStart, dete
ctedCount);

                                        displayingInCommandPrompt(disk, detectedStart,
detectedCount);
                                }
                        }
                baseFileNumber++;
                }
        }
}
void displayingInCommandPrompt( struct ATRSSDISK *disk, int start, int count){

        int i, j, biggest;
        biggest = start + count;
        int counter = 0;

        for (i = start-1; i < biggest-1; i++){
                for (j = 0; j < 125; j++){
                        if(isprint(disk->sector[i][j])){
                                printf("%c", disk->sector[i][j]);
                                counter++;
                        }else if (disk->sector[i][j] == 0x00a){
                                printf("\n");
                        }
                }
        }
```

```
}

void savingToLocal(FILE *fp, struct ATRSSDISK *disk, int start, int count){

        fp = fopen("/cs/home/yusufadi/Desktop/eecs2031/lab03/JAB.txt", "w+");

        int i, j, biggest;
        biggest = start + count;

        for (i = start-1; i < biggest-1; i++){
                for (j = 0; j < 125; j++){
                        if(isprint(disk->sector[i][j])){
                                fprintf(fp, "%c", disk->sector[i][j]);
                        }else if (disk->sector[i][j] == 0x00a){
                                fprintf(fp, "\n");
                        }
                }
        }

        fclose(fp);
}


int checkingFile(char name[], char input[]){
        int i, count;
        count = 0;
        for(i = 0; input[i] != '\0'; i++){
                if(input[i] == name[i]){
                        count++;
                }
        }
        if(count > 2){
        return 1;
        }else{
        return 0;
        }
}

int checkingFileExt(char ext[], char input[]){

        int i, finger, count;
        count = 0;
        finger = 0;
        for(i = 0; input[i] != '\0'; i++){
                if(input[i] == ext[finger]){
                        count++;
                        finger++;
                }
        }
        if(count > 0){
        return 1;
        }else{
        return 0;
        }
}



int main(int argc, char *argv[])
{
  struct ATRSSDISK *disk;

  if(argc != 3) {
```

```
    fprintf(stderr,"usage: %s disk\n", argv[0]);
    exit(1);
  }
  if((disk = readDisk(argv[1])) == (struct ATRSSDISK *)NULL) {
    fprintf(stderr,"Unable to read disk %s\n", argv[1]);
    exit(1);
  }
  extract(stdout, disk, argv[2]); /* put it in atari offset notation 1..720 */
  freeDisk(disk);
  return 0;

}
```