Date: 10/5/2019

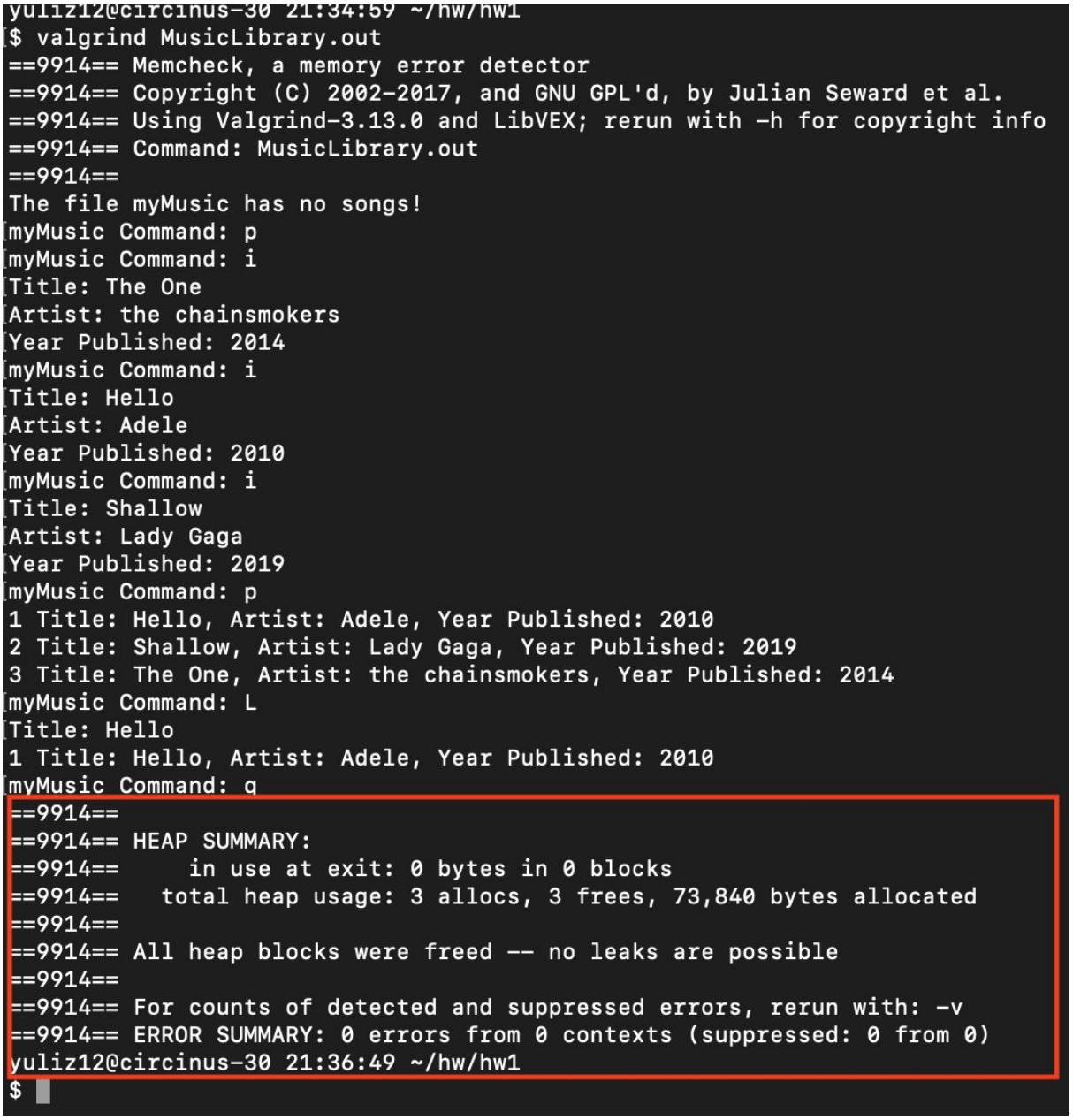
CS253P Homework 1

Yulin Zhang, ID: 43765138

Compile Time:

```
yuliz12@circinus-30 21:38:12 ~/hw/hw1
$ make
g++ -ggdb -std=c++11 -Wpedantic -Wall -Wextra -Werror -Wzero-as-null-pointer
-constant MusicLibrary.cpp -o MusicLibrary.out
yuliz12@circinus-30 21:38:16 ~/hw/hw1
$ MusicLibrary.out
myMusic Command: i
Title: Hello
Artist: Adele
Year Published: 2018
myMusic Command: Call you Mine
myMusic Command: i
Title: Call you Mine
Artist: the chainsmokers
Year Published: 2019
myMusic Command: p
1 Title: Call you Mine, Artist: the chainsmokers, Year Published: 2019
2 Title: Hello, Artist: Adele, Year Published: 2018
3 Title: Hello, Artist: Adele, Year Published: 2010
4 Title: Shallow, Artist: Lady Gaga, Year Published: 2019
5 Title: The One, Artist: the chainsmokers, Year Published: 2014
myMusic Command: q
yuliz12@circinus-30 21:39:05 ~/hw/hw1
$
```

Valgrind Run:



Code Highlight:

```c
struct Song music_library[SONGMAX];
int current_number_of_songs = 0;
const char *fileName = NULL;

//open file and return with FILE*
FILE * open_file(const char *fileName, const char *mode) {
    FILE *fp = fopen(fileName, mode);
    if(fp==NULL) {
        fprintf(stderr, "Can't open %s!\n", fileName);
        exit(1);
    }
    return fp;
}

//swap two songs in the music library
void swap(int i, int j) {
    Song tmpSong_1, tmpSong_2;

    strcpy(tmpSong_1.title, music_library[i].title);
    strcpy(tmpSong_1.artist, music_library[i].artist);
    tmpSong_1.year_published = music_library[i].year_published;

    strcpy(tmpSong_2.title, music_library[j].title);
    strcpy(tmpSong_2.artist, music_library[j].artist);
    tmpSong_2.year_published = music_library[j].year_published;

    strcpy(music_library[i].title, tmpSong_2.title);
    strcpy(music_library[i].artist, tmpSong_2.artist);
    music_library[i].year_published = tmpSong_2.year_published;

    strcpy(music_library[j].title, tmpSong_1.title);
    strcpy(music_library[j].artist, tmpSong_1.artist);
    music_library[j].year_published = tmpSong_1.year_published;
}
```

```c
//convert string to lowercase
void strlwr(char *str)
{
  char *pointer = (char *)str;
  while (*pointer) {
      *pointer = tolower((char)*pointer);
      pointer++;
  }
}

//compare two songs by their titles
int compare_song_titles(char *title_1, char *title_2) {
    char tmp_title_1[STRMAX], tmp_title_2[STRMAX];
    strcpy(tmp_title_1, title_1);
    strcpy(tmp_title_2, title_2);
    strlwr(tmp_title_1);
    strlwr(tmp_title_2);
    return strcmp(tmp_title_1, tmp_title_2);
}

//partition in quicksort
int partition (int low, int high) {
    int pivot = high;
    int i = low - 1;
    for (int j = low; j <= high- 1; j++) {
        if (compare_song_titles(music_library[j].title, music_library[pivot].title) < 0) {
            i++;
            swap(i, j);
        }
    }
    swap(i+1, high);
    return i + 1;
}

//quicksort algorithm
void quick_sort(int low, int high) {
    if (low < high) {
        int pivot = partition(low, high);
        quick_sort(low, pivot - 1);
        quick_sort(pivot + 1, high);
    }
}

//sort music library by title
void sort_MusicLibrary() {
    int low = 0, high = current_number_of_songs-1;
    quick_sort(low, high);
}
```

```c
//parse a song info
void load_MusicLibrary(char *buf, int current_number_of_songs) {
    char *token = strtok(buf,",");
    if(token!=NULL) {
        strcpy(music_library[current_number_of_songs].title, token);
    }
    token = strtok(NULL,",");
    if(token!=NULL) {
        strcpy(music_library[current_number_of_songs].artist, token);
    }
    token = strtok(NULL,"\n");
    if(token!=NULL) {
        music_library[current_number_of_songs].year_published = atoi(token);
    }
}

//write a song to file
void write_song(FILE *ofp, int index) {
    fprintf(ofp, "%s,", music_library[index].title);
    fprintf(ofp, "%s,", music_library[index].artist);
    fprintf(ofp, "%d\n", music_library[index].year_published);
}

//read songs from file
void read_song(FILE *ifp) {
    fseek (ifp, 0, SEEK_END);
    int size = int(ftell(ifp));
    if(size!=0) {
        fseek(ifp, 0, SEEK_SET);
        char buf[256];
        while(fgets(buf, sizeof(buf), ifp)) {
            load_MusicLibrary(buf, current_number_of_songs);
            current_number_of_songs++;
        }
        sort_MusicLibrary();
    } else {
        printf("The file %s has no songs!\n", fileName);
    }
    fclose(ifp);
}
```

```c
//auxiliary function, called by function remove_song_from_MusicLibrary_by_name()
void crunch_up_from_index(int index) {
    while(index<current_number_of_songs) {
        strcpy(music_library[index].title, music_library[index+1].title);
        strcpy(music_library[index].artist, music_library[index+1].artist);
        music_library[index].year_published = music_library[index+1].year_published;
        index++;
    }
}

//auxiliary function, not used
void crunch_down_from_index(int index) {
    int cur_index = current_number_of_songs-1;
    while(cur_index>index) {
        strcpy(music_library[cur_index].title, music_library[cur_index-1].title);
        strcpy(music_library[cur_index].artist, music_library[cur_index-1].artist);
        music_library[cur_index].year_published = music_library[cur_index-1].year_published;
        cur_index--;
    }
}

//binary search algorithm
int binary_search(int low, int high, char *song_title) {
    char tmp_song_title[STRMAX];
    strcpy(tmp_song_title, song_title);
    while(low<=high) {
        int mid = low + (high - low)/2;
        int num = compare_song_titles(song_title, music_library[mid].title);
        if(num == 0) {
            return mid;
        } else if(num < 0) {
            high = mid - 1;
        } else if(num > 0) {
            low = mid + 1;
        }
    }
    return -1;
}
```

```
//find index by calling function binary_search()
int find_index_of_song_with_name(char *song_title) {
    int low = 0, high = current_number_of_songs-1;
    return binary_search(low, high, song_title);
}

//add song to the music library
void add_song_to_MusicLibrary(char *song_title, char *song_artist, int year_published) {
    int index = current_number_of_songs;
    strcpy(music_library[index].title, song_title);
    strcpy(music_library[index].artist, song_artist);
    music_library[index].year_published = year_published;
    current_number_of_songs++;
    sort_MusicLibrary();
}

//check if the input of year is a valid, called by function input_song_info()
bool is_numeric(char *pointer) {
    int length = int(strlen(pointer)), i = 0;
    while (i<length) {
        char ch = (char)pointer[i++];
        if('0'>ch || '9'<ch) {
            return false;
        }
    }
    return true;
}

//input song information, related with command "I"
void input_song_info() {
    char song_title[STRMAX], song_artist[STRMAX], year_published[STRMAX];
    string tmpStr;
    printf("Title: ");
    scanf(" %[^\n]s", song_title);
    printf("Artist: ");
    scanf(" %[^\n]s", song_artist);
    printf("Year Published: ");
    scanf(" %[^\n]s", year_published);
    while(!is_numeric(year_published)) {
        printf("Year Published: ");
        scanf(" %[^\n]s", year_published);
    }
    int year = atoi(year_published);
    add_song_to_MusicLibrary(song_title, song_artist, year);
}
```

```c
//when look up, there may be some songs with same titles
void check_titles_to_left(char *song_title, int index) {
    int low = 0, cur = index - 1;
    while(cur>=low) {
        if(compare_song_titles(song_title, music_library[cur].title) == 0) {
            printf("%d Title: %s, Artist: %s, Year Published: %d\n", cur+1, music_library[cur].title,
                music_library[cur].artist, music_library[cur].year_published);
            cur--;
        } else {
            break;
        }
    }
}

//when look up, there may be some songs with same titles
void check_titles_to_right(char *song_title, int index) {
    int high = current_number_of_songs - 1, cur = index + 1;
    while(cur<=high) {
        if(compare_song_titles(song_title, music_library[cur].title) ==0) {
            printf("%d Title: %s, Artist: %s, Year Published: %d\n", cur+1, music_library[cur].title,
                music_library[cur].artist, music_library[cur].year_published);
            cur++;
        } else {
            break;
        }
    }
}

//look up a song in the library by calling function find_index_of_song_with_name()
void look_up_song_in_MusicLibrary() {
    char song_title[STRMAX];
    printf("Title: ");
    scanf(" %[^\n]s", song_title);
    int index = find_index_of_song_with_name(song_title);
    if(index==-1) {
        printf("%s is not found in the music library!\n", song_title);
    } else {
        printf("%d Title: %s, Artist: %s, Year Published: %d\n", index+1, music_library[index].title,
            music_library[index].artist, music_library[index].year_published);
        check_titles_to_left(song_title, index);
        check_titles_to_right(song_title, index);
    }
}
```

```
//evaluate a command
bool evaluate_command(char *command) {
    bool isQuit = false;
    int length = int(strlen(command));
    if(length!=1) {
        return isQuit;
    }
    char com = toupper(command[0]);
    switch(com) {
        case 'I':
            input_song_info(); break;
        case 'P':
            print_MusicLibrary(true); break;
        case 'D':
            remove_song_from_MusicLibrary_by_name(); break;
        case 'L':
            look_up_song_in_MusicLibrary(); break;
        case 'Q':
            print_MusicLibrary(false);
            isQuit = true;
    }
    return isQuit;
}

//read command from user input
void read_command() {
    printf("myMusic Command: ");
    char command[STRMAX];
    scanf(" %[^\n]s", command);
    while(!evaluate_command(command)) {
        printf("myMusic Command: ");
        scanf(" %[^\n]s", command);
    }
}

//the starting point of the program
int main(int argc, const char *argv[]) {
    fileName = argc==2? argv[1]:"myMusic";
    FILE *ifp = open_file(fileName, "a+");
    read_song(ifp);
    read_command();
    return 0;
}
```

Edge Cases:

1. Output of the program on standard input.

```
yuliz12@circinus-30 21:59:13 ~/hw/hw1
$ g++ MusicLibrary.cpp -o MusicLibrary
yuliz12@circinus-30 21:59:23 ~/hw/hw1
$ MusicLibrary
The file myMusic has no songs!
myMusic Command: i
Title: C my title
Artist: C my artist
Year Published: 2014
myMusic Command: i
Title: A my title
Artist: A my artist
Year Published: 2014
myMusic Command: i
Title: B my title
Artist: B my artist
Year Published: 2014
myMusic Command: p
1 Title: A my title, Artist: A my artist, Year Published: 2014
2 Title: B my title, Artist: B my artist, Year Published: 2014
3 Title: C my title, Artist: C my artist, Year Published: 2014
myMusic Command: L
Title: B my title
2 Title: B my title, Artist: B my artist, Year Published: 2014
myMusic Command: D
Title: B my title
B my title has been deleted successfully!
myMusic Command: P
1 Title: A my title, Artist: A my artist, Year Published: 2014
2 Title: C my title, Artist: C my artist, Year Published: 2014
myMusic Command: Q
yuliz12@circinus-30 22:00:44 ~/hw/hw1
$ cat myMusic
A my title,A my artist,2014
C my title,C my artist,2014
yuliz12@circinus-30 22:00:50 ~/hw/hw1
$
```

2. This program would have a corner case when **more than two arguments are given in the command line. it will simply open the default file “myMusic”.**

```
yuliz12@circinus-30 22:07:35 ~/hw/hw1
$ MusicLibrary music myAlbum songs
myMusic Command: p
1 Title: A my title, Artist: A my artist, Year Published: 2014
2 Title: C my title, Artist: C my artist, Year Published: 2014
myMusic Command: q
yuliz12@circinus-30 22:08:08 ~/hw/hw1
$ ls
Makefile  MusicLibrary*  MusicLibrary.cpp  MusicLibrary.out*  myMusic
yuliz12@circinus-30 22:08:16 ~/hw/hw1
$ cat myMusic
A my title,A my artist,2014
C my title,C my artist,2014
yuliz12@circinus-30 22:08:24 ~/hw/hw1
$
```

3. This program would have a corner case when **the input of myMusic Command isn't valid.**

```
yuliz12@circinus-30 22:09:43 ~/hw/hw1
$ MusicLibrary
myMusic Command: iekjsekse
myMusic Command: ppp
myMusic Command: qqq
myMusic Command: 1223
myMusic Command: pdql
myMusic Command: p
1 Title: A my title, Artist: A my artist, Year Published: 2014
2 Title: C my title, Artist: C my artist, Year Published: 2014
myMusic Command: r
myMusic Command:
```

4. This program would have a corner case when **the file given is empty.**

```
yuliz12@circinus-30 22:12:54 ~/hw/hw1
$ cat myMusic
A my title,A my artist,2014
C my title,C my artist,2014
yuliz12@circinus-30 22:13:14 ~/hw/hw1
$ rm myMusic
rm: remove regular file 'myMusic'? Y
removed 'myMusic'
yuliz12@circinus-30 22:13:26 ~/hw/hw1
$ MusicLibrary
The file myMusic has no songs!
myMusic Command: p
myMusic Command: q
yuliz12@circinus-30 22:13:39 ~/hw/hw1
$ cat myMusic
yuliz12@circinus-30 22:13:46 ~/hw/hw1
$ MusicLibrary
The file myMusic has no songs!
myMusic Command: i
Title: Roses
Artist: The chainsmokers
Year Published: 2019
myMusic Command: p
1 Title: Roses, Artist: The chainsmokers, Year Published: 2019
myMusic Command: q
yuliz12@circinus-30 22:14:29 ~/hw/hw1
$ cat myMusic
Roses,The chainsmokers,2019
yuliz12@circinus-30 22:14:40 ~/hw/hw1
$
```

5. This program would have a corner case when **the year of song isn't a number.**

```
yuliz12@circinus-30 22:17:19 ~/hw/hw1
$ MusicLibrary
myMusic Command: i
Title: CS253P UCI mcs
Artist: california state
Year Published: year
Year Published: 2019year
Year Published: year2019
Year Published: 2019
myMusic Command: p
1 Title: CS253P UCI mcs, Artist: california state, Year Published: 2019
2 Title: Roses, Artist: The chainsmokers, Year Published: 2019
myMusic Command:
```

6. This program would have a corner case when **looking up for a song, it will print all songs with the same title ignoring letter cases.**

```
myMusic Command: p
1 Title: hello my love, Artist: westlife, Year Published: 2018
2 Title: Hello My Love, Artist: Westlife, Year Published: 2019
3 Title: My Love, Artist: my love, Year Published: 2001
4 Title: my love, Artist: westlife, Year Published: 2018
5 Title: Thunder, Artist: Imagine Dragons, Year Published: 2016
6 Title: thunder, Artist: imaginedragons, Year Published: 2016
myMusic Command: L
Title: my love
3 Title: My Love, Artist: my love, Year Published: 2001
4 Title: my love, Artist: westlife, Year Published: 2018
myMusic Command: l
Title: thunder
5 Title: Thunder, Artist: Imagine Dragons, Year Published: 2016
6 Title: thunder, Artist: imaginedragons, Year Published: 2016
myMusic Command: l
Title: hello my love
1 Title: hello my love, Artist: westlife, Year Published: 2018
2 Title: Hello My Love, Artist: Westlife, Year Published: 2019
myMusic Command:
```

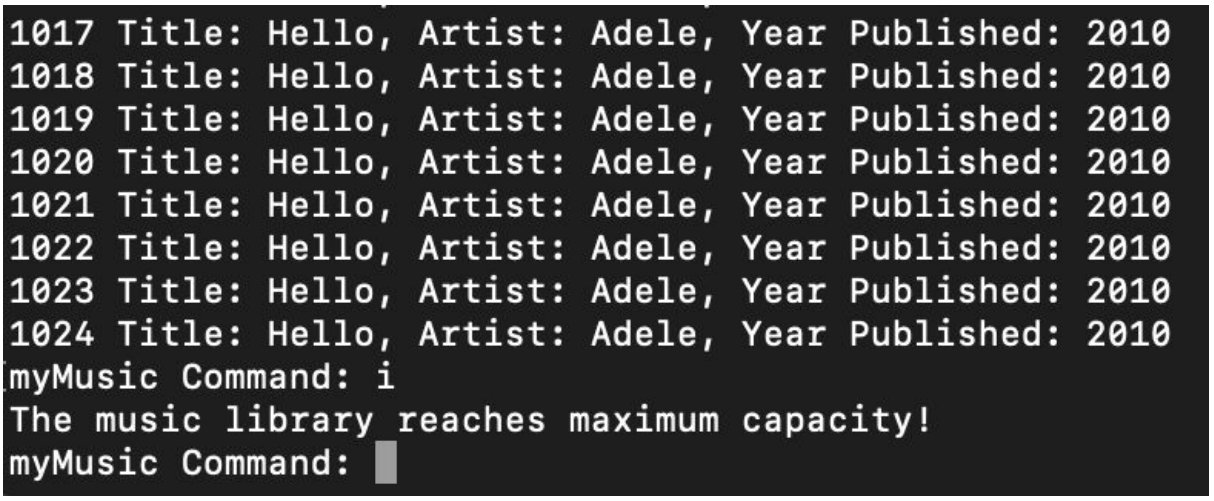
7. This program would have a corner case when **looking up for or removing a song, the title input by user isn't found in the music library.**

```
myMusic Command: p
1 Title: hello my love, Artist: westlife, Year Published: 2018
2 Title: Hello My Love, Artist: Westlife, Year Published: 2019
3 Title: My Love, Artist: my love, Year Published: 2001
4 Title: my love, Artist: westlife, Year Published: 2018
5 Title: Thunder, Artist: Imagine Dragons, Year Published: 2016
6 Title: thunder, Artist: imaginedragons, Year Published: 2016
myMusic Command: L
Title: my
my is not found in the music library!
myMusic Command: D
Title: warriors
warriors is not found in the music library!
myMusic Command: D
Title: my love
my love has been deleted successfully!
myMusic Command: p
1 Title: hello my love, Artist: westlife, Year Published: 2018
2 Title: Hello My Love, Artist: Westlife, Year Published: 2019
3 Title: my love, Artist: westlife, Year Published: 2018
4 Title: Thunder, Artist: Imagine Dragons, Year Published: 2016
5 Title: thunder, Artist: imaginedragons, Year Published: 2016
```

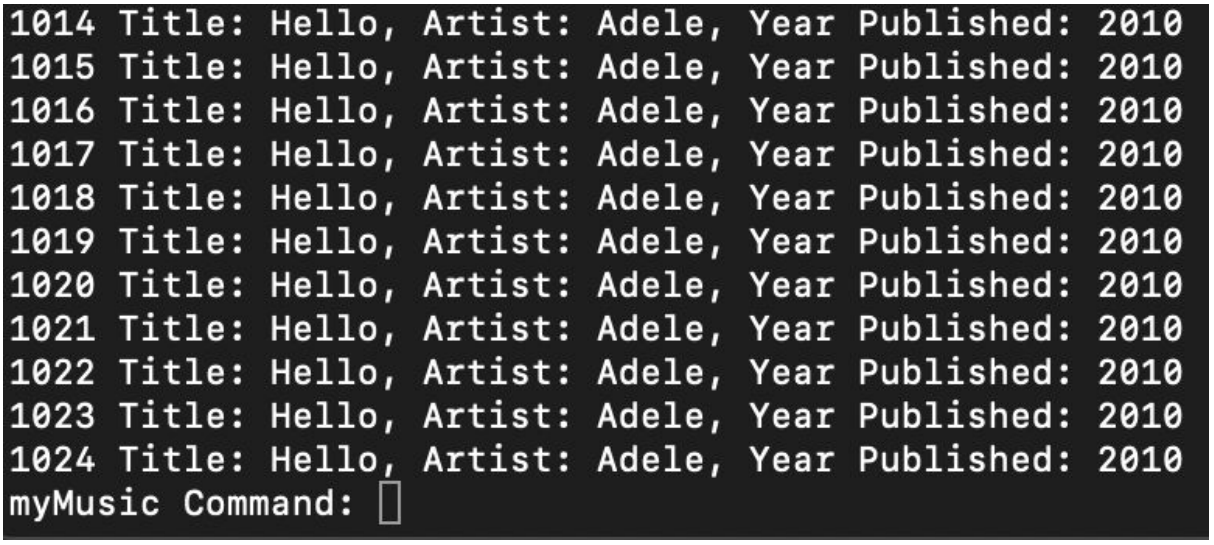
8. This program would have a corner case when **swapping two songs, Valgrind will prompt "Source and destination overlap in strcpy(char*, char*), because both char* point to the same address, which is not allowed in Valgrind. And the way to solve it is to use an extra pair of char strings.**

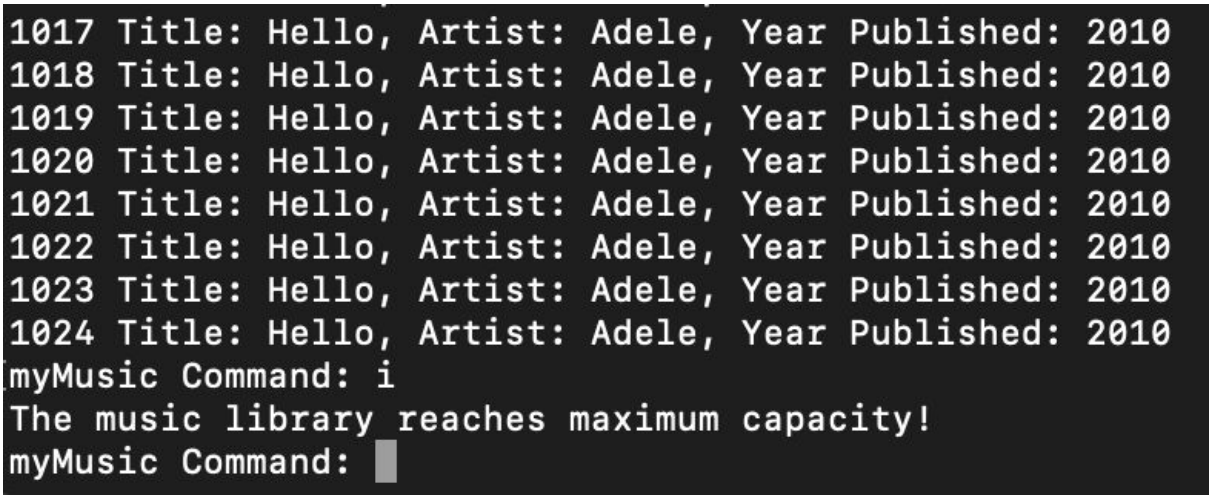
```
Makefile  MusicLibrary.cpp  MusicLibrary.out*  myMusic
yuliz12@circinus-30 20:22:25 ~/hw/hw1
$ valgrind MusicLibrary.out
==1348== Memcheck, a memory error detector
==1348== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==1348== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==1348== Command: MusicLibrary.out
==1348==
==1348== Source and destination overlap in strcpy(0x603100, 0x603100)
==1348==    at 0x4C2CB42: strcpy (vg_replace_strmem.c:510)
==1348==    by 0x400D4E: swap(int, int) (MusicLibrary.cpp:39)
==1348==    by 0x400F70: partition(int, int) (MusicLibrary.cpp:75)
==1348==    by 0x400FB5: quick_sort(int, int) (MusicLibrary.cpp:85)
==1348==    by 0x401009: sort_MusicLibrary() (MusicLibrary.cpp:94)
==1348==    by 0x40121E: read_song(_IO_FILE*) (MusicLibrary.cpp:131)
==1348==    by 0x401A79: main (MusicLibrary.cpp:316)
==1348==
==1348== Source and destination overlap in strcpy(0x603164, 0x603164)
==1348==    at 0x4C2CB42: strcpy (vg_replace_strmem.c:510)
==1348==    by 0x400D93: swap(int, int) (MusicLibrary.cpp:40)
==1348==    by 0x400F70: partition(int, int) (MusicLibrary.cpp:75)
==1348==    by 0x400FB5: quick_sort(int, int) (MusicLibrary.cpp:85)
==1348==    by 0x401009: sort_MusicLibrary() (MusicLibrary.cpp:94)
==1348==    by 0x40121E: read_song(_IO_FILE*) (MusicLibrary.cpp:131)
==1348==    by 0x401A79: main (MusicLibrary.cpp:316)
==1348==
==1348== Source and destination overlap in strcpy(0x603298, 0x603298)
==1348==    at 0x4C2CB42: strcpy (vg_replace_strmem.c:510)
==1348==    by 0x400D4E: swap(int, int) (MusicLibrary.cpp:39)
==1348==    by 0x400F88: partition(int, int) (MusicLibrary.cpp:78)
==1348==    by 0x400FB5: quick_sort(int, int) (MusicLibrary.cpp:85)
==1348==    by 0x401009: sort_MusicLibrary() (MusicLibrary.cpp:94)
==1348==    by 0x40121E: read_song(_IO_FILE*) (MusicLibrary.cpp:131)
==1348==    by 0x401A79: main (MusicLibrary.cpp:316)
==1348==
==1348== Source and destination overlap in strcpy(0x6032fc, 0x6032fc)
==1348==    at 0x4C2CB42: strcpy (vg_replace_strmem.c:510)
==1348==    by 0x400D93: swap(int, int) (MusicLibrary.cpp:40)
==1348==    by 0x400F88: partition(int, int) (MusicLibrary.cpp:78)
==1348==    by 0x400FB5: quick_sort(int, int) (MusicLibrary.cpp:85)
==1348==    by 0x401009: sort_MusicLibrary() (MusicLibrary.cpp:94)
==1348==    by 0x40121E: read_song(_IO_FILE*) (MusicLibrary.cpp:131)
==1348==    by 0x401A79: main (MusicLibrary.cpp:316)
==1348==
```

9. This program would have a corner case when **trying to insert a new song, but the music library has already reached maximum capacity, so that no more songs can be added.**



10. This program would have a corner case when **the number of songs read from the given file exceeds the maximum number that the music library can hold. Then the program will only load the maximum number of songs to the music library.**





Here is the file.

myMusic

```
1004  Hello,Adele,2010
1005  Hello,Adele,2010
1006  Hello,Adele,2010
1007  Hello,Adele,2010
1008  Hello,Adele,2010
1009  Hello,Adele,2010
1010  Hello,Adele,2010
1011  Hello,Adele,2010
1012  Hello,Adele,2010
1013  Hello,Adele,2010
1014  Hello,Adele,2010
1015  Hello,Adele,2010
1016  Hello,Adele,2010
1017  Hello,Adele,2010
1018  Hello,Adele,2010
1019  Hello,Adele,2010
1020  Hello,Adele,2010
1021  Hello,Adele,2010
1022  Hello,Adele,2010
1023  Hello,Adele,2010
1024  Hello,Adele,2010
1025  Hello,Adele,2010
1026  Hello,Adele,2010
1027  Hello,Adele,2010
1028  Hello,Adele,2010
1029  Hello,Adele,2010
1030  Hello,Adele,2010
1031  Hello,Adele,2010
1032  Hello,Adele,2010
1033  Hello,Adele,2010
1034  Hello,Adele,2010
```