

```

1 #ifndef HEADER_H_
2 #define HEADER_H_
3 #include <iostream>
4 #include <string>
5 #include <fstream>
6 #include <iomanip>
7 #include <unistd.h>
8
9 struct DVD {
10     std::string title;
11     std::string leadActor;
12     std::string subActor;
13     std::string genre;
14     std::string altGenre;
15     int year;
16     int rating;
17     std::string synopsis;
18     DVD *nextNode;
19 };
20 enum MenuOptions { Exit = 0, OutputEntireList, TitleSearch, GenreSearch,
ActorSearch, YearSearch, RatingSearch};
21 void heading(); //outputs heading
22 DVD *readInput(); //reads data from file and stores it in heap using linked
list
23 void dispMenu(); //displays menu
24 MenuOptions switchValidation(); //validates switch input and returns enum
MenuOptions
25 void searchKeyLogic(
26     DVD *head,
27     std::fstream &oFile,
28     const int &key,
29     int &index); //searches for year or rating
30
31 void searchKeyLogic(
32     DVD *head,
33     std::fstream &oFile,
34     const std::string &key,
35     int &index); //searches for genre or actor
36
37 void outputList(
38     DVD *head,
39     std::fstream &oFile,
40     int &index); //outputs entire list to outfile
41
42 //testing which format of prototype is more used if you can give me some help
43
44 void multiMoviePrint(std::fstream &oFile, DVD* node, int &index); //prints
multi movies
45 void switchOption(const MenuOptions &option, DVD* head, std::fstream
&oFile); //directs user on there choice
46 void subString(DVD *head); //formats data
47 void genreSearch(DVD* head, std::fstream &oFile, int &index); //genre search
output
48 void actorSearch(DVD* head, std::fstream &oFile, int &index); //actor search
output
49 void yearSearch(DVD* head, std::fstream &oFile, int &index); //year search
output
50 void ratingSearch(DVD* head, std::fstream &oFile, int &index); //rating search
output

```

```
51 void yearValidation(int &keyInt);//validates year using try catch
52 void ratingValidation(int &keyInt);//validates rating using try catch
53 void printSingleMovie(DVD* node, std::fstream &oFile);//prints single movie
54 void titleSearch(DVD* head, std::fstream &oFile);//title search output
55 void titleSearchLogic(DVD* head, std::fstream &oFile, std::string
    strKey);//searches linked list for title
56 void deallocate(DVD* head);//dallocates memory
57
58 #endif
```

```

1  /*****
2  * AUTHOR      : Carlos Aguilera
3  * STUDENT ID  : 1152562
4  * LAB #3     : Searching Linked Lists
5  * CLASS      : CS1B
6  * SECTION    : M-W
7  * DUE DATE   : 04.05.22
8  *****/
9
10 #include "../include/header.h"
11
12 /*****
13 * Title: Searching Linked Lists
14 * -----
15 * PROGRAM:
16 *   This Program pulls data from an input file that is specified or it will
17 *   resort to a default, it is using linked lists to collect that data and
18 *   store it in the heap. The functionality of the program should be as follows
19 *   user starts application and selects file to read from then that file if
20 *   valid gets collected as a linked list. Then you can obtain that data and
21 *   search through it using sequential search.
22 * -----
23 * Data Table
24 * -----
25 * std::string outputFile IN - used to create or open output file
26 * DVD* head IN, CALC & OUT - creates pointer that points to DVD struct
27 * MenuOptions option IN & CALC - enum that holds what option they choose
28 * std::fstream oFile OUT - used for outputting to a file
29 *****/
30
31 int main() {
32     heading();
33     std::string outputFile;
34     DVD *head = NULL;
35     head = readInput();//reads input file
36     subString(head);
37
38     MenuOptions option;
39     std::fstream oFile;
40     std::cout << "Which output file would you like to use(type d for default
41 file)? ";
42     std::getline(std::cin, outputFile);
43     if (outputFile == "d")
44         outputFile = "output.txt";
45     oFile.open(outputFile, std::ios::out);//opens file to write
46
47     do {
48         dispMenu();//displays menu
49         option = switchValidation();//validates switch choice before choosing
50         switchOption(option, head, oFile);
51     } while(option != Exit);
52
53     oFile.close();
54     deallocate(head);
55     head = NULL;
56     return 0;
57 }

```

```

1 #include "../include/header.h"
2 /*****
3  * Title: dispMenu
4  * -----
5  * FUNCTION:
6  *   This function displays the menu
7  * -----
8  * NO Data Table
9  * -----
10 *****/
11
12 void dispMenu() {
13     std::cout << "\nDVD MENU OPTIONS\n\n";
14     std::cout << "1 - Output Entire List\n";
15     std::cout << "2 - Title Search\n";
16     std::cout << "3 - Genre search\n";
17     std::cout << "4 - Actor Search\n";
18     std::cout << "5 - Year Search\n";
19     std::cout << "6 - Rating Search (0 - 9)\n";
20     std::cout << "0 - EXIT\n";
21 }

```

```

1 #include "../include/header.h"
2 /*****
3  * Title: readInput
4  * -----
5  * FUNCTION:
6  *   This function reads from the input file specified, how it goes about it
7  * is as so creates head ptr to DVD and creates node ptr to DVD but allocates
8  * a new DVD on the heap. What happens is node = new DVD and head = NULL
9  * and first iteration of the while loop the next node equals head which is
10 * NULL what that is doing is setting the node to be NULL and last in the
11 * linked list. head then has the address of original node and node = new DVD
12 * -----
13 * Data Table
14 * -----
15 * DVD* head = NULL IN & CALC - used in linked list
16 * DVD* node = new DVD IN & CALC - used in linked list
17 * std::fstream inFile IN - used to grab movie data
18 * std::string temp IN - used to grab \n
19 * std::string inputFile IN & CALC - used to identify input file
20 *****/
21
22
23 DVD *readInput() {
24     DVD *head = NULL;
25     DVD *node = new DVD;
26     std::fstream inFile;
27     std::string temp;
28     std::string inputFile;
29     do {
30         std::cout << "Which input file would you like to use(type d for default
file)? ";
31         std::getline(std::cin, inputFile);
32         if (inputFile != "d")
33             inputFile = "../build/" + inputFile;//adding folder structure to file
34         else
35             inputFile = "../build/AS5-BigInFile.txt";//default
36
37         inFile.open(inputFile, std::ios::in);
38
39         if (inFile.is_open()) {
40             while (!inFile.eof()) {while not end of file
41                 std::getline(inFile, node -> title);
42                 std::getline(inFile, node -> leadActor);
43                 std::getline(inFile, node -> subActor);
44                 std::getline(inFile, node -> genre);
45                 std::getline(inFile, node -> altGenre);
46                 inFile >> node -> year >> node -> rating;
47                 inFile.ignore(10000, '\n');
48                 std::getline(inFile, node -> synopsis);
49                 std::getline(inFile, temp);
50
51                 node->nextNode = head;
52                 head = node;
53                 node = new DVD;
54             }
55             inFile.close();
56         }else
57             std::cout << "Enter Valid File name\n";// if no input file exist
58     } while (head == NULL);//run while havent been initialized

```

```
59 | delete node;  
60 | return head;  
61 | }
```

```

1 #include "../include/header.h"
2 /*****
3  * Title: subString
4  * -----
5  * FUNCTION:
6  *   This function formats the data to match all calculations made to it
7  * -----
8  * NO Data Table
9  * -----
10 *****/
11
12 void subString(DVD* head) {
13     DVD* node = head;
14     while (node->nextNode != NULL) {
15         //cuts one character off the ends of each string
16         node->title = node->title.substr(0, node->title.size() - 1);
17         node->leadActor = node->leadActor.substr(0, node->leadActor.size() - 1);
18         node->subActor = node->subActor.substr(0, node->subActor.size() - 1);
19         node->genre = node->genre.substr(0, node->genre.size() - 1);
20         node->altGenre = node->altGenre.substr(0, node->altGenre.size() - 1);
21         node->synopsis = node->synopsis.substr(0, node->synopsis.size() - 1);
22         node = node->nextNode;
23     }
24 }

```

```

1 #include "../include/header.h"
2 /*****
3  * Title: validateInput
4  * -----
5  * FUNCTION:
6  *   This is an input validation function it handles all major validations
7  *   from the switch statement, year, and rating validation
8  * -----
9  * Data Table
10 * -----
11 * bool inputValidated = false; CALC - to determine if input was validated
12 * int choice; IN & CALC - input choice and verify if its valid
13 *****/
14
15 MenuOptions switchValidation() {
16     int choice;
17     bool inputValidated = false;
18     //try catch wrapped in a do while until input is validated
19     do {
20         try {
21             std::cout << "Enter an option (0 to exit): ";
22             std::cin >> choice;
23             if (std::cin.fail()) { //if cin fails throw exception
24                 std::cin.clear(); //clears buffer for next cin
25                 std::cin.ignore(10, '\n');
26                 throw(true);
27             }
28             inputValidated = true;
29         }
30         catch(bool invalid) {
31             std::cout << "Please enter a number!\n";
32         }
33     } while (!inputValidated);
34
35     return static_cast <MenuOptions> (choice); //returns int but static cast to
enum MenuOptions
36 }
37
38 void yearValidation(int &keyInt) {
39     bool inputValidated = false;
40     do {
41         try {
42             std::cout << "\nWhich year are you looking for? ";
43             std::cin >> keyInt;
44             if (std::cin.fail()) {
45                 std::cin.clear();
46                 std::cin.ignore(1000, '\n');
47                 throw(true);
48             } else if (keyInt >= 1878 && keyInt <= 3000) //if key is valid and its
between years then
49                 inputValidated = true;
50             else {
51                 std::cout << "The number " << keyInt << " is an invalid entry\n";
52                 std::cout << "**** Please input a number between 1878 and 3000
****\n";
53             }
54         }
55         catch(bool invalid) {

```



```

56         std::cout << "**** Please input a NUMBER between 1878 and 3000
****\n";
57     }
58     } while (!inputValidated);
59 }
60
61 void ratingValidation(int &keyInt) {
62     bool inputValidated = false;
63     do {
64         try {
65             std::cout << "\nWhich rating are you looking for? ";
66             std::cin >> keyInt;
67             if (std::cin.fail()) {
68                 std::cin.clear();
69                 std::cin.ignore(1000, '\n');
70                 throw(true);
71             } else if (keyInt >= 0 && keyInt <= 9)
72                 inputValidated = true;
73             else {
74                 std::cout << "The number " << keyInt << " is an invalid entry\n";
75                 std::cout << "**** Please input a number between 0 and 9 ****\n";
76             }
77         }
78         catch(bool invalid) {
79             std::cout << "**** Please input a NUMBER between 0 and 9 ****\n";
80         }
81     } while (!inputValidated);
82 }

```

```

1 #include "../include/header.h"
2 /*****
3  * Title: switchOption
4  * -----
5  * FUNCTION:
6  *   Switch function handles the choice of user
7  * -----
8  * Data Table
9  * -----
10 * int index CALC – used to determine index of linked list
11 *****/
12
13 void switchOption(const MenuOptions &option, DVD* head, std::fstream &oFile)
14 {
15     int index = 0;
16     switch (option) {
17         case OutputEntireList:
18             std::cout << "\nCOMPLETE MOVIE LISTING!\n\n";
19             oFile << "All Movies Found:\n";
20             outputList(head, oFile, index);
21             oFile << "\n";
22             break;
23         case TitleSearch:
24             titleSearch(head, oFile);
25             break;
26         case GenreSearch:
27             genreSearch(head, oFile, index);
28             break;
29         case ActorSearch:
30             actorSearch(head, oFile, index);
31             break;
32         case YearSearch:
33             yearSearch(head, oFile, index);
34             break;
35         case RatingSearch:
36             ratingSearch(head, oFile, index);
37             break;
38         case Exit:
39             std::cout << "\nThank You!!\n";
40             break;
41         default:
42             std::cout << "Enter A Valid Option\n";
43             break;
44     }
45 }
46 /*****
47  * Title: titleSearch
48  * -----
49  * FUNCTION:
50  *   handles the output for title search
51  * -----
52  * Data Table
53  * -----
54  * std::strKey IN & CALC input key and searches linked list for it
55 *****/
56
57 void titleSearch(DVD* head, std::fstream &oFile) {
58     std::string strKey;

```

```

59
60     std::cout << "Which Title are you looking for? ";
61     std::cin.ignore(1000, '\n');
62     std::getline(std::cin, strKey);
63
64     titleSearchLogic(head, oFile, strKey);
65 }
66 /*****
67  * Title: genreSearch
68  * -----
69  * FUNCTION:
70  *     Handles the output for genreSearch
71  * -----
72  * Data Table
73  * -----
74  * std::string strKey IN & CALC - holds user genre input
75  *****/
76
77 void genreSearch(DVD* head, std::fstream &oFile, int &index) {
78     std::string strKey;
79
80     std::cout << "\nWhich Genre are you looking for? ";
81     std::cin.ignore(1000, '\n');
82     std::getline(std::cin, strKey);
83     std::cout << "\nSearching for the genre " << strKey << "\n";
84
85     searchKeyLogic(head, oFile, strKey, index);
86
87     if (index == 0) //if after going through the linked list the index still is
88         0 then movie was not found
89         std::cout << "Sorry, no movies for the genre " << strKey << " were
90         found.\n";
91     else {
92         std::cout << "Found " << index << " movies for the genre " << strKey <<
93         "\n";
94         oFile << "\n";
95     }
96 }
97 /*****
98  * Title: actorSearch
99  * -----
100 * FUNCTION:
101 *     handles actor search output
102 * -----
103 * Data Table
104 * -----
105 * std::string strKey IN & CALC - holds actor name
106 *****/
107
108 void actorSearch(DVD* head, std::fstream &oFile, int &index) {
109     std::string strKey;
110
111     std::cout << "\nWhich Actor are you looking for? ";
112     std::cin.ignore(1000, '\n');
113     std::getline(std::cin, strKey);
114     std::cout << "\nSearching for the actor " << strKey << "\n";
115
116     searchKeyLogic(head, oFile, strKey, index);
117
118     if (index == 0)

```

```

116     std::cout << "Sorry, no movies for the actor " << strKey << " were
found.\n";
117     else {
118         std::cout << "Found " << index << " movies for the actor " << strKey <<
"!\\n";
119         oFile << "\\n";
120     }
121 }
122 /*****
123 * Title: yearSearch
124 * -----
125 * FUNCTION:
126 *     handles year search output
127 * -----
128 * Data Table
129 * -----
130 * int keyInt IN & CALC - user year input and then we validate it
131 *****/
132
133 void yearSearch(DVD* head, std::fstream &oFile, int &index) {
134     int keyInt;
135     yearValidation(keyInt);
136
137     std::cout << "\\nSearching for the year " << keyInt << "\\n";
138     searchKeyLogic(head, oFile, keyInt, index);
139
140     if (index == 0)
141         std::cout << "Sorry, no movies for the year " << keyInt << " were
found.\n";
142     else {
143         std::cout << "Found " << index << " movies for the year " << keyInt <<
"!\\n";
144         oFile << "\\n";
145     }
146 }
147 /*****
148 * Title: ratingSearch
149 * -----
150 * FUNCTION:
151 *     handles rating search output
152 * -----
153 * Data Table
154 * -----
155 * int keyInt IN & CALC - user rating input and then we validate it
156 *****/
157
158 void ratingSearch(DVD* head, std::fstream &oFile, int &index) {
159     int keyInt;
160     ratingValidation(keyInt);
161
162     std::cout << "\\nSearching for the rating " << keyInt << "\\n";
163     searchKeyLogic(head, oFile, keyInt, index);
164
165     if (index == 0)
166         std::cout << "Sorry, no movies for the rating " << keyInt << " were
found.\n";
167     else {
168         std::cout << "Found " << index << " movies for the rating " << keyInt
<< "!\\n";
169         oFile << "\\n";

```

```

170     }
171 }
172 /*****
173  * Title: outputList
174  * -----
175  * FUNCTION:
176  *     handles entire output of movies
177  * -----
178  * NO Data Table
179  * -----
180 *****/
181
182 void outputList(DVD *head, std::fstream &oFile, int &index) {
183     DVD *node = head;
184     while (node->nextNode != NULL) {
185
186         multiMoviePrint(oFile, node, index);
187         node = node->nextNode;
188     }
189 }

```

```

1 #include "../include/header.h"
2 /*****
3  * Title: titleSearchLogic
4  * -----
5  * FUNCTION:
6  *   handles the title search logic has a while loop the iterates until
7  * end of linked list or until found
8  * -----
9  * Data Table
10 * -----
11 * DVD* node = head CALC - use it not to mess with head
12 * bool found CALC - false until found
13 *****/
14
15 void titleSearchLogic(DVD* head, std::fstream &oFile, std::string strKey) {
16     DVD *node = head;
17     bool found = false;
18     while (node->nextNode != NULL && !found) {
19
20         if (node->title == strKey) {//current strKey being a title of course
21             std::cout << "Found the movie " << strKey << "!\n";
22             printSingleMovie(node, oFile);
23             found = true;
24         }
25
26         node = node->nextNode;
27         if (node->nextNode == NULL)//output statement if not found
28             std::cout << "Sorry, the movie \" " << strKey << " \" was not
found.\n";
29     }
30 }
31 /*****
32  * Title: searchKeyLogic
33  * -----
34  * FUNCTION:
35  *   These are 2 overloaded functions that handle searches for year, rating
36  * genre, and lead actor
37  * -----
38  * NO Data Table
39  * -----
40 *****/
41
42 void searchKeyLogic(DVD *head, std::fstream &oFile, const int &key, int
&index) {
43     DVD *node = head;
44     while (node->nextNode != NULL) {
45
46         if (node->year == key) {
47             if (index == 0)
48                 oFile << "Search by year for " << key << " found:\n";
49             multiMoviePrint(oFile, node, index);
50         } else if (node->rating == key) {
51             if (index == 0)
52                 oFile << "Search by rating for " << key << " found:\n";
53             multiMoviePrint(oFile, node, index);
54         }
55
56         node = node->nextNode;
57     }

```

```
58 }
59 void searchKeyLogic(DVD *head, std::fstream &oFile, const std::string &key,
60 int &index) {
61     DVD *node = head;
62     while (node->nextNode != NULL) {
63         if (node->genre == key || node->altGenre == key) {
64             if (index == 0)
65                 oFile << "Search by genre for " << key << " found:\n";
66             multiMoviePrint(oFile, node, index);
67         } else if (node->leadActor == key || node->subActor == key) {
68             if (index == 0)
69                 oFile << "Search by actor for " << key << " found:\n";
70             multiMoviePrint(oFile, node, index);
71         }
72         node = node->nextNode;
73     }
74 }
75 }
```

```

1 #include "../include/header.h"
2 /*****
3  * Title: printSingleMovie
4  * -----
5  * FUNCTION:
6  *   Handles word wrap functionality and output to file a single movie
7  * -----
8  * Data Table
9  * -----
10 * std::string line OUT - used to output line of text
11 * std::string word CALC - used to hold word and add to line
12 * const int maxLineLength CALC max length a line can be
13 *****/
14
15 void printSingleMovie(DVD* node, std::fstream &oFile) {
16     std::string line;
17     std::string word;
18     const int maxLineLength = 75;
19     oFile << std::left;
20     oFile <<
21     "*****\n";
22     oFile << "Title: " << node->title << "\n";
23     oFile << "-----\n";
24     oFile << "Year: " << node->year << " " << "Rating: " << node->rating
25     << "\n";
26     oFile << "-----\n";
27     oFile << std::setw(18) << "Leading Actor:" << std::setw(25) << node-
28     >leadActor << "Genre 1: " << node->genre << "\n";
29     oFile << "Supporting Actor: " << std::setw(25) << node->subActor << "Genre
30     2: " << node->altGenre << "\n";
31     oFile << "-----\n";
32     oFile << "PLOT:\n";
33     for (int i = 0; i < node->synopsis.length(); i++) { //logic for word wrap
34         //runs character by character if its a space then its ignored if not its
35         added to word once word is completed it adds it to line
36         //if line length is greater than max then we output line
37         if (node->synopsis.at(i) != ' ')
38             word.push_back(node->synopsis.at(i));
39         else if (word.length() + line.length() > maxLineLength) {
40             oFile << line << "\n";
41             line.clear();
42             line += word;
43             word.clear();
44         } else {
45             line += word + ' ';
46             word.clear();
47         }
48         if (i + 1 == node->synopsis.length()) {
49             line += word;
50             oFile << line << "\n";
51         }
52     }
53     oFile <<
54     "*****\n";
55     oFile << "\n";

```



```
49 | oFile << std::right;  
50 | }
```

```

1 #include "../include/header.h"
2 /*****
3  * Title: multiMoviePrint
4  * -----
5  * FUNCTION:
6  *   handles output of multiple movies also handles greater than format
7  *   amount feature and cuts it accordingly
8  * -----
9  * NO Data Table
10 * -----
11 *****/
12
13 void multiMoviePrint(std::fstream &oFile, DVD* node, int &index) {
14     ++index;
15     oFile << std::left;
16     if (index <= 10) {
17         if(index == 1) {
18             oFile << "MOVIE #" << std::setw(50) << " TITLE" << "YEAR " <<
19 "RATING " << std::setw(18) << "GENRE" << std::setw(18) << "ALT GENRE" <<
20 std::setw(20) << "LEAD ACTOR" << "SUPPORTING ACTOR\n";
21             oFile << "-----\n";
22         }
23         oFile << " " << std::setw(6) << index << std::setw(48);
24
25         if (node->title.size() > 47) { //if title is greater than format space
26             oFile << (node->title.substr(0, 44)) + "..."; //we cut and add ...
27         }else
28             oFile << node->title;
29
30         oFile << std::setw(8) << node->year << std::setw(5) << node->rating <<
31 std::setw(18) << node->genre << std::setw(18) << node->altGenre <<
32 std::setw(20);
33
34         if (node->leadActor.size() > 18) {
35             oFile << (node->leadActor.substr(0, 15)) + "...";
36         }else
37             oFile << node->leadActor;
38
39         if (node->subActor.size() > 18) {
40             oFile << (node->subActor.substr(0, 15)) + "...";
41         }else
42             oFile << node->subActor;
43         oFile << "\n";
44         if (index == 10)
45             oFile << "... \n";
46     }
47     oFile << std::right;
48 }

```

```
1 #include "../include/header.h"
2
3 void deallocate(DVD* head) {
4     DVD* node = head;
5     while(node != NULL) {
6         head = node->nextNode;
7         delete node;
8         node = head;
9     }
10 }
```

```

1 #include "../include/header.h"
2
3 void heading() {
4
5     /*****
6      * CONSTANTS
7      * -----
8      * OUTPUT - USED FOR CLASS HEADING
9      * -----
10     * PROGRAMMER : Programmer's Name
11     * CLASS      : Student's Course
12     * SECTION    : Class Days and Times
13     * LAB_NUM    : Lab Number (specific to this lab)
14     * LAB_NAME   : Title of the Lab
15     *****/
16     const char PROGRAMMER[] = "Carlos Aguilera";
17     const char CLASS[]      = "CS1B";
18     const char SECTION[]    = "MW: 7:30p - 9:50p";
19     const int LAB_NUM       = 3;
20     const char LAB_NAME[]   = "Searching Linked Lists";
21
22     std::cout << std::left;
23     std::cout << "*****\n";
24     std::cout << "*   PROGRAMMED BY : " << PROGRAMMER << std::endl;
25     std::cout << "*   " << std::setw(14) << "CLASS" << ": " << CLASS <<
std::endl;
26     std::cout << "*   " << std::setw(14) << "SECTION" << ": " << SECTION <<
std::endl;
27     std::cout << "*   LAB #" << std::setw(9) << LAB_NUM << ": " << LAB_NAME <<
std::endl;
28     std::cout << "*****\n\n";
29     std::cout << std::right;
30 }

```

1 All Movies Found:

MOVIE #	TITLE	YEAR	RATING	GENRE	
	ALT GENRE	LEAD ACTOR	SUPPORTING ACTOR		
1	007 – Casino Royale		2006	8	Action
	Adventure	Daniel Craig	Eva Green		
2	007 – Quantum of Solace		2008	7	Action
	Adventure	Daniel Craig	Olga Kurylenko		
3	10 Items or Less		2006	7	Comedy
	Drama	Morgan Freeman	Paz Vega		
4	15 minutes		2001	6	Action
	Crime	Robert De Niro	Edward Burns		
5	17 Again		2009	7	Comedy
	Comedy	Zac Effron	Leslie Mann		
6	21		2008	7	Drama
	Drama	Jim Sturgess	Kevin Spacey		
7	25th Hour		2002	8	Drama
	Drama	Edward Norton	Philip Seymour ...		
8	3:10 to Yuma		2007	8	Drama
	Crime	Russell Crowe	Christian Bale		
9	50 First Dates		2004	7	Romantic
Comedy	Comedy	Adam Sandler	Drew Barrymore		
10	88 Minutes		2007	6	Action
	Crime	Al Pacino	Alicia Witt		

14 ...

16 \*\*\*\*\*

17 Title: Men in Black

19 Year: 1997      Rating: 7

21 Leading Actor:    Tommy Lee Jones      Genre 1: Comedy

22 Supporting Actor: Will Smith      Genre 2: Comedy

24 PLOT:

25 Two men who keep an eye on aliens in New York City must try to save the  
26 worldafter the aliens threaten to blow it up.

27 \*\*\*\*\*

29 Search by genre for Romantic Comedy found:

MOVIE #	TITLE	YEAR	RATING	GENRE	
	ALT GENRE	LEAD ACTOR	SUPPORTING ACTOR		
1	50 First Dates		2004	7	Romantic
Comedy	Comedy	Adam Sandler	Drew Barrymore		
2	Alfie		2004	6	Romantic
Comedy	Comedy	Jude Law	Renee Taylor		
3	Always		1989	6	Romantic
Comedy	Romance	Richard Dryfus	Holly Hunter		
4	American President, The		1995	7	Romantic
Comedy	Comedy	Michael Douglas	Annette Bening		
5	Benny & Joon		1993	7	Romantic
Comedy	Comedy	Mary Stuart Mas...	Johnny Depp		
6	Break-up, The		2006	6	Romantic
Comedy	Romantic Comedy	Jennifer Aniston	Vince Vaughn		
7	Bridget Jones 2: The Edge of Reason		2004	6	Romantic
Comedy	Drama	Renee Zellweger	Colin Firth		

39	8	Bridget Jones's Diary w. BJ2	2001	7	Romantic
	Comedy	Drama	Renee Zellweger	Hugh Grant	
40	9	Casanova	2005	7	Romantic
	Comedy	Comedy	Heath Ledger	Oliver Platt	
41	10	Couples Retreat	2009	6	Romantic
	Comedy	Comedy	Vince Vaughn	Jon Favreau	
42	...				

44 Search by actor for Anthony Hopkins found:

45	MOVIE #	TITLE	YEAR	RATING	GENRE
		ALT GENRE	LEAD ACTOR	SUPPORTING ACTOR	
46	-----				
47	1	Bobby	2006	7	
	Biography	Drama	Anthony Hopkins	Harry Belafonte	
48	2	Legends of the Fall	1995	7	Drama
		Drama	Brad Pitt	Anthony Hopkins	
49	3	Meet Joe Black	1998	7	Drama
		Drama	Brad Pitt	Anthony Hopkins	
50	4	Proof	2005	7	Drama
		Drama	Gweneth Paltrow	Anthony Hopkins	
51	5	Shadowlands	1994	7	
	Biography	Drama	Anthony Hopkins	Debra Winger	
52	6	World's Fastest Indian, The	2005	8	
	Biography	Drama	Anthony Hopkins	Iain Rea	

54 Search by year for 2007 found:

55	MOVIE #	TITLE	YEAR	RATING	GENRE
		ALT GENRE	LEAD ACTOR	SUPPORTING ACTOR	
56	-----				
57	1	3:10 to Yuma	2007	8	Drama
		Crime	Russell Crowe	Christian Bale	
58	2	88 Minutes	2007	6	Action
		Crime	Al Pacino	Alicia Witt	
59	3	A Mighty Heart	2007	7	
	Biography	Drama	Dan Futterman	Angelina Joile	
60	4	Across the Universe	2007	8	Musical
		Drama	Evan Rachel Wood	Jim Sturgess	
61	5	August Rush	2007	8	Drama
		Music	Freddie Highmore	Keri Russell	
62	6	Before the Devil Knows You're Dead	2007	7	Drama
		Crime	Philip Seymour ...	Ethan Hawke	
63	7	Bourne Ultimatum	2007	8	Action
		Adventure	Matt Damon	Joan Allen	
64	8	Breach	2007	7	
	Biography	Crime	Chris Cooper	Ryan Phillippe	
65	9	Breakfast with Scot	2007	7	Comedy
		Drama	Cameron Anzel	Benz Antoine	
66	10	Charlie Wilson's War	2007	7	
	Biography	Drama	Tom Hanks	Julia Roberts	
67	...				

69 Search by rating for 8 found:

70	MOVIE #	TITLE	YEAR	RATING	GENRE
		ALT GENRE	LEAD ACTOR	SUPPORTING ACTOR	
71	-----				
72	1	007 - Casino Royale	2006	8	Action
		Adventure	Daniel Craig	Eva Green	

73	2	25th Hour		2002	8	Drama
		Drama	Edward Norton	Philip Seymour ...		
74	3	3:10 to Yuma		2007	8	Drama
		Crime	Russell Crowe	Christian Bale		
75	4	A Beautiful Mind		2001	8	
		Biography	Drama	Russell Crowe	Ed Harris	
76	5	Across the Universe		2007	8	Musical
		Drama	Evan Rachel Wood	Jim Sturgess		
77	6	August Rush		2007	8	Drama
		Music	Freddie Highmore	Keri Russell		
78	7	Back to the Future (1)		1985	8	Action
		Comedy	Michael J. Fox	Christopher Lloyd		
79	8	Bank Job, The		2008	8	Drama
		Crime	Jason Straham	Saffron Burrows		
80	9	Batman Begins		2005	8	Action
		Action	Christian Bale	Michael Caine		
81	10	Beauty and the Beast		1991	8	
		Family/Animation	Animation	Paige O'Hara	Robby Benson	
82	...					
83						
84						

```
1 Which input file would you like to use(type d for default file)? d
2 Which output file would you like to use(type d for default file)? d
3
4 DVD MENU OPTIONS
5
6 1 - Output Entire List
7 2 - Title Search
8 3 - Genre search
9 4 - Actor Search
10 5 - Year Search
11 6 - Rating Search (0 - 9)
12 0 - EXIT
13 Enter an option (0 to exit): 1
14
15 COMPLETE MOVIE LISTING!
16
17
18 DVD MENU OPTIONS
19
20 1 - Output Entire List
21 2 - Title Search
22 3 - Genre search
23 4 - Actor Search
24 5 - Year Search
25 6 - Rating Search (0 - 9)
26 0 - EXIT
27 Enter an option (0 to exit): 0
28
29 Thank You!!
30 > ./main
31 Which input file would you like to use(type d for default file)? d
32 Which output file would you like to use(type d for default file)? d
33
34 DVD MENU OPTIONS
35
36 1 - Output Entire List
37 2 - Title Search
38 3 - Genre search
39 4 - Actor Search
40 5 - Year Search
41 6 - Rating Search (0 - 9)
42 0 - EXIT
43 Enter an option (0 to exit): 1
44
45 COMPLETE MOVIE LISTING!
46
47
48 DVD MENU OPTIONS
49
50 1 - Output Entire List
51 2 - Title Search
52 3 - Genre search
53 4 - Actor Search
54 5 - Year Search
55 6 - Rating Search (0 - 9)
56 0 - EXIT
57 Enter an option (0 to exit): 2
58 Which Title are you looking for? Men in Black
59 Found the movie Men in Black!
```



```
60
61 DVD MENU OPTIONS
62
63 1 - Output Entire List
64 2 - Title Search
65 3 - Genre search
66 4 - Actor Search
67 5 - Year Search
68 6 - Rating Search (0 - 9)
69 0 - EXIT
70 Enter an option (0 to exit): 2
71 Which Title are you looking for? Shawshank Redemption
72 Sorry, the movie " Shawshank Redemption " was not found.
73
74 DVD MENU OPTIONS
75
76 1 - Output Entire List
77 2 - Title Search
78 3 - Genre search
79 4 - Actor Search
80 5 - Year Search
81 6 - Rating Search (0 - 9)
82 0 - EXIT
83 Enter an option (0 to exit): 3
84
85 Which Genre are you looking for? Romantic Comedy
86
87 Searching for the genre Romantic Comedy
88 Found 45 movies for the genre Romantic Comedy!
89
90 DVD MENU OPTIONS
91
92 1 - Output Entire List
93 2 - Title Search
94 3 - Genre search
95 4 - Actor Search
96 5 - Year Search
97 6 - Rating Search (0 - 9)
98 0 - EXIT
99 Enter an option (0 to exit): 3
100
101 Which Genre are you looking for? HORROR
102
103 Searching for the genre HORROR
104 Sorry, no movies for the genre HORROR were found.
105
106 DVD MENU OPTIONS
107
108 1 - Output Entire List
109 2 - Title Search
110 3 - Genre search
111 4 - Actor Search
112 5 - Year Search
113 6 - Rating Search (0 - 9)
114 0 - EXIT
115 Enter an option (0 to exit): 4
116
117 Which Actor are you looking for? Anthony Hopkins
118
119 Searching for the actor Anthony Hopkins
```

```
120 Found 6 movies for the actor Anthony Hopkins!
121
122 DVD MENU OPTIONS
123
124 1 - Output Entire List
125 2 - Title Search
126 3 - Genre search
127 4 - Actor Search
128 5 - Year Search
129 6 - Rating Search (0 - 9)
130 0 - EXIT
131 Enter an option (0 to exit): 4
132
133 Which Actor are you looking for? Don Johnson
134
135 Searching for the actor Don Johnson
136 Sorry, no movies for the actor Don Johnson were found.
137
138 DVD MENU OPTIONS
139
140 1 - Output Entire List
141 2 - Title Search
142 3 - Genre search
143 4 - Actor Search
144 5 - Year Search
145 6 - Rating Search (0 - 9)
146 0 - EXIT
147 Enter an option (0 to exit): 5
148
149 Which year are you looking for? 2007
150
151 Searching for the year 2007
152 Found 52 movies for the year 2007!
153
154 DVD MENU OPTIONS
155
156 1 - Output Entire List
157 2 - Title Search
158 3 - Genre search
159 4 - Actor Search
160 5 - Year Search
161 6 - Rating Search (0 - 9)
162 0 - EXIT
163 Enter an option (0 to exit): 5
164
165 Which year are you looking for? 1800
166 The number 1800 is an invalid entry
167 **** Please input a number between 1878 and 3000 ****
168
169 Which year are you looking for? 5
170 The number 5 is an invalid entry
171 **** Please input a number between 1878 and 3000 ****
172
173 Which year are you looking for? 3001
174 The number 3001 is an invalid entry
175 **** Please input a number between 1878 and 3000 ****
176
177 Which year are you looking for? a
178 **** Please input a NUMBER between 1878 and 3000 ****
179
```

```
180 Which year are you looking for? 1900
181
182 Searching for the year 1900
183 Sorry, no movies for the year 1900 were found.
184
185 DVD MENU OPTIONS
186
187 1 - Output Entire List
188 2 - Title Search
189 3 - Genre search
190 4 - Actor Search
191 5 - Year Search
192 6 - Rating Search (0 - 9)
193 0 - EXIT
194 Enter an option (0 to exit): 6
195
196 Which rating are you looking for? -1
197 The number -1 is an invalid entry
198 **** Please input a number between 0 and 9 ****
199
200 Which rating are you looking for? 10
201 The number 10 is an invalid entry
202 **** Please input a number between 0 and 9 ****
203
204 Which rating are you looking for? z
205 **** Please input a NUMBER between 0 and 9 ****
206
207 Which rating are you looking for? 8
208
209 Searching for the rating 8
210 Found 136 movies for the rating 8!
211
212 DVD MENU OPTIONS
213
214 1 - Output Entire List
215 2 - Title Search
216 3 - Genre search
217 4 - Actor Search
218 5 - Year Search
219 6 - Rating Search (0 - 9)
220 0 - EXIT
221 Enter an option (0 to exit): -1
222 Enter A Valid Option
223
224 DVD MENU OPTIONS
225
226 1 - Output Entire List
227 2 - Title Search
228 3 - Genre search
229 4 - Actor Search
230 5 - Year Search
231 6 - Rating Search (0 - 9)
232 0 - EXIT
233 Enter an option (0 to exit): 7
234 Enter A Valid Option
235
236 DVD MENU OPTIONS
237
238 1 - Output Entire List
239 2 - Title Search
```

```
240 3 - Genre search
241 4 - Actor Search
242 5 - Year Search
243 6 - Rating Search (0 - 9)
244 0 - EXIT
245 Enter an option (0 to exit): a
246 Please enter a number!
247 Enter an option (0 to exit): 0
248
249 Thank You!!
```