

```
1 *****
2 *   PROGRAMMED BY : Andrew Gharios
3 *   STUDENT ID    : 1449366
4 *   CLASS         : M-Th 5-7:20p
5 *   ASSIGNMENT #5 : OPP - DVD Movie List
6 *****
7 *****
8 Title: Antwone Fisher
9 -----
10 Year: 2002          Rating: 7
11 -----
12 Leading Actor:     Denzel Washington      Genre 1: Biography
13 Supporting Actor:  Derek Luke             Genre 2: Drama
14 -----
15 PLOT:
16 Antwone Fisher, a young navy man, is forced to see a psychiatrist after a
17 violent outburst against a fellow crewman. During the course of treatment a
18 painful past is revealed and a new hope begins.
19 *****
20
21
22 *****
23 Title: Bagger Vance, The Legend of
24 -----
25 Year: 2000          Rating: 6
26 -----
27 Leading Actor:     Will Smith              Genre 1: Drama
28 Supporting Actor:  Matt Damon              Genre 2: Drama
29 -----
30 PLOT:
31 A down-and-out golfer attempts to recover his game and his life with help
32 from a mystical caddy
33 *****
34
35
36 *****
37 Title: Cinderella Man
38 -----
39 Year: 2005          Rating: 8
40 -----
41 Leading Actor:     Russell Crowe           Genre 1: Biography
42 Supporting Actor:  Renee Zellweger         Genre 2: Drama
43 -----
44 PLOT:
45 The story of James Braddock, a supposedly washed up boxer who came back to
46 become a champion and an inspiration in the 1930s.
47 *****
48
49
```

```
50 *****
51 Title: Five People You Meet in Heaven, The
52 -----
53 Year: 2004          Rating: 7
54 -----
55 Leading Actor:      Jon Voight          Genre 1: Drama
56 Supporting Actor:   Ellen Burstyn       Genre 2: Drama
57 -----
58 PLOT:
59 On his 83rd birthday, Eddie (Voight), a war vet and a maintenance worker at
60 the Ruby Pier amusement park, dies while trying to save a girl who is
61 sitting under a falling ride. When he awakens in the afterlife, he
62 encounters five people with ties to his corporeal existence who help him
63 understand the meaning of his life.
64 *****
65
66
67 *****
68 Title: Good Night, Good Luck
69 -----
70 Year: 2005          Rating: 8
71 -----
72 Leading Actor:      David Strathairn     Genre 1: Biography
73 Supporting Actor:   George Clooney       Genre 2: Drama
74 -----
75 PLOT:
76 In the early 1950's, the threat of Communism created an air of paranoia in
77 the United States and exploiting those fears was Senator Joseph McCarthy of
78 Wisconsin. However, CBS reporter Edward R. Murrow and his producer Fred W.
79 Friendly decided to take a stand and challenge McCarthy and expose him for
80 the fear monger he was. However, their actions took a great personal toll
81 on both men, but they stood by their convictions and helped to bring down
82 one of the most controversial senators in American history.
83 *****
84
85
86 *****
87 Title: Invictus
88 -----
89 Year: 2009          Rating: 8
90 -----
91 Leading Actor:      Morgan Freeman       Genre 1: Biography
92 Supporting Actor:   Matt Damon           Genre 2: Drama
93 -----
94 PLOT:
95 Nelson Mandela, in his first term as the South African President, initiates
96 a unique venture to unite the apartheid-torn land: enlist the national
97 rugby team on a mission to win the 1995 Rugby World Cup.
98 *****
```

99  
100  
101 \*\*\*\*\*  
102 Title: Miss Pettigrew Lives for a Day  
103 -----  
104 Year: 2008            Rating: 7  
105 -----  
106 Leading Actor:     Frances McDormand            Genre 1: Comedy  
107 Supporting Actor: Amy Adams                    Genre 2: Romance  
108 -----  
109 PLOT:  
110 Guinevere Pettigrew, a middle-aged London governess, finds herself unfairly  
111 dismissed from her job. An attempt to gain new employment catapults her  
112 into the glamorous world and dizzying social whirl of an American actress  
113 and singer, Delysia Lafosse.  
114 \*\*\*\*\*  
115  
116  
117 \*\*\*\*\*  
118 Title: Pay it Forward  
119 -----  
120 Year: 2000            Rating: 7  
121 -----  
122 Leading Actor:     Kevin Spacey                Genre 1: Drama  
123 Supporting Actor: Helen Hunt                   Genre 2: Romance  
124 -----  
125 PLOT:  
126 Like some other kids, 12-year-old Trevor McKinney believed in the goodness  
127 of human nature. Like many other kids, he was determined to change the  
128 world for the better. Unlike most other kids, he succeeded.  
129 \*\*\*\*\*  
130  
131  
132 \*\*\*\*\*  
133 Title: Pianist, The  
134 -----  
135 Year: 2003            Rating: 9  
136 -----  
137 Leading Actor:     Adrien Brody                Genre 1: Biography  
138 Supporting Actor: Thomas Kreschmann           Genre 2: Drama  
139 -----  
140 PLOT:  
141 A Polish Jewish musician struggles to survive the destruction of the Warsaw  
142 ghetto of World War II.  
143 \*\*\*\*\*  
144  
145  
146 \*\*\*\*\*  
147 Title: Pirates of the Caribbean 1: The Curse of the Bl...

```
-----
148 -----
149 Year: 2003          Rating: 8
150 -----
151 Leading Actor:      Johnny Depp          Genre 1: Action
152 Supporting Actor:   Orlando Bloom        Genre 2: Action
153 -----
154 PLOT:
155 This swash-buckling tale follows the quest of Captain Jack Sparrow, a savvy
156 pirate, and Will Turner, a resourceful blacksmith, as they search for
157 Elizabeth Swann. Elizabeth, the daughter of the governor and the love of
158 Will's life, has been kidnapped by the feared Captain Barbossa. Little do
159 they know, but the fierce and clever Barbossa has been cursed. He, along
160 with his large crew, is under an ancient curse, doomed for eternity to
161 neither live, nor die. That is, unless a blood sacrifice is made. Full of
162 edge-of-the-seat action and swashbuckling adventures, this is a movie you
163 won't want to miss!
164 *****
165
166
167 *****
168 Title: Pursuit of Happyness
169 -----
170 Year: 2008          Rating: 8
171 -----
172 Leading Actor:      Will Smith           Genre 1: Biography
173 Supporting Actor:   Jaden Smith          Genre 2: Drama
174 -----
175 PLOT:
176 A struggling salesman takes custody of his son as he's poised to begin a
177 life-changing professional endeavor.
178 *****
179
180
181 *****
182 Title: Rent
183 -----
184 Year: 2005          Rating: 7
185 -----
186 Leading Actor:      Anthony Rapp         Genre 1: Musical
187 Supporting Actor:   Adam Pascal          Genre 2: Drama
188 -----
189 PLOT:
190 This is the film version of the Pulitzer and Tony Award winning musical
191 about Bohemians in the East Village of New York City struggling with life,
192 love and AIDS, and the impacts they have on America.
193 *****
194
195
196 *****
```

197 Title: Secondhand Lions  
198 -----  
199 Year: 2003            Rating: 8  
200 -----  
201 Leading Actor:     Haley Joel Osmet            Genre 1: Comedy  
202 Supporting Actor: Robert Duvall            Genre 2: Comedy  
203 -----  
204 PLOT:  
205 A coming-of-age story about a shy, young boy sent by his irresponsible  
206 mother to spend the summer with his wealthy, eccentric uncles in Texas.  
207 \*\*\*\*\*  
208  
209  
210 \*\*\*\*\*  
211 Title: Shackelton  
212 -----  
213 Year: 2002            Rating: 8  
214 -----  
215 Leading Actor:     Kenneth Branagh            Genre 1: Biography  
216 Supporting Actor: John Grillo            Genre 2: Drama  
217 -----  
218 PLOT:  
219 The true story of Shackleton's 1914 Endurance expedition to the the South  
220 Pole and his epic struggle to lead his 28 man crew to safety after his ship  
221 was crushed in the pack ice.  
222 \*\*\*\*\*  
223  
224  
225 \*\*\*\*\*  
226 Title: Shawshank Redemption, The  
227 -----  
228 Year: 1994            Rating: 9  
229 -----  
230 Leading Actor:     Tim Robbins            Genre 1: Drama  
231 Supporting Actor: Morgan Freeman            Genre 2: Crime  
232 -----  
233 PLOT:  
234 Two imprisoned men bond over a number of years, finding solace and eventual  
235 redemption through acts of common decency.  
236 \*\*\*\*\*  
237  
238  
239 \*\*\*\*\*  
240 Title: Thelma and Louise  
241 -----  
242 Year: 1991            Rating: 7  
243 -----  
244 Leading Actor:     Susan Sarandon            Genre 1: Action  
245 Supporting Actor: Geena Davis            Genre 2: Action

-----  
PLOT:  
Whilst on a short weekend getaway, Louise shoots a man who had tried to rape Thelma. Due to the incriminating circumstances, they make a run for it but are soon followed closely by the authorities including a local policeman who is sympathetic to their plight. The federal authorities, however, have less compassion and thus a cross country chase ensues for the two fugitives. Along the way, both women rediscover the strength of their friendship and surprising aspects of their personalities and self-strengths in the trying times.  
\*\*\*\*\*  
-----  
\*\*\*\*\*  
Title: Unfinished Life, An  
-----  
Year: 2005                      Rating: 7  
-----  
Leading Actor:        Robert Redford                      Genre 1: Drama  
Supporting Actor: Jennifer Lopez                      Genre 2: Drama  
-----  
PLOT:  
A down on her luck woman, desperate to provide care for her daughter, moves in with her father in-law from whom she is estranged. Through time, they learn to forgive each other and heal old wounds.  
\*\*\*\*\*  
-----  
\*\*\*\*\*  
Title: Whip It  
-----  
Year: 2009                      Rating: 7  
-----  
Leading Actor:        Ellen Page                      Genre 1: Drama  
Supporting Actor: Marcia Gay Harden                      Genre 2: Sport  
-----  
PLOT:  
In Bodeen, Texas, an indie-rock loving misfit finds a way of dealing with her small-town misery after she discovers a roller derby league in nearby Austin.  
\*\*\*\*\*  
-----  
\*\*\*\*\*  
Title: X-Men  
-----  
Year: 2000                      Rating: 7  
-----  
Leading Actor:        Hugh Jackman                      Genre 1: Action

295 Supporting Actor: Patrick Stewart                      Genre 2: Action  
296 -----  
297 PLOT:  
298 All over the planet, unusual children are born with an added twist to their  
299 genetic code. This X-factor allows the children to perform extraordinary  
300 feats - flight, telekinetics, laser beams from the eyes and more. One Dr.  
301 Charles Xavier gathers the children to a place where he can train them to  
302 use their powers for themselves and the forces of good he dubs these  
303 children his X-men and hijinks ensue.  
304 \*\*\*\*\*  
305  
306  
307 \*\*\*\*\*  
308 Title: Yankee Doodle Dandee  
309 -----  
310 Year: 1942                      Rating: 8  
311 -----  
312 Leading Actor:     James Cagney                      Genre 1: Musical  
313 Supporting Actor: Joan Leslie                      Genre 2: Biography  
314 -----  
315 PLOT:  
316 This film depicts the life of the renowned musical composer, playwright,  
317 actor, dancer and singer George M. Cohan.  
318 \*\*\*\*\*  
319  
320  
321

```
1 *****
2 *   PROGRAMMED BY : Andrew Gharios
3 *   STUDENT ID    : 1449366
4 *   CLASS         : M-Th 5-7:20p
5 *   ASSIGNMENT #5 : OPP - DVD Movie List
6 *****
7 Which input file would you like to use? inFile.txt
8 Which output file would you like to use? OutFile.txt
9
10 Name: Yankee Doodle Dandee
11 Name: X-Men
12 Name: Whip It
13 Name: Unfinished Life, An
14 Name: Thelma and Louise
15 Name: Shawshank Redemption, The
16 Name: Shackelton
17 Name: Secondhand Lions
18 Name: Rent
19 Name: Pursuit of Happyness
20 Name: Pirates of the Caribbean 1: The Curse of the Black Pearl
21 Name: Pianist, The
22 Name: Pay it Forward
23 Name: Miss Petigrew Lives for a Day
24 Name: Invictus
25 Name: Good Night, Good Luck
26 Name: Five People You Meet in Heaven, The
27 Name: Cinderella Man
28 Name: Bagger Vance, The Legend of
29 Name: Antwone Fisher
30
31 Writing output file OutFile.txt
```



```
1  #ifndef HEADER_H_
2  #define HEADER_H_
3
4  #include <iostream> // cin, cout.
5  #include <string>   // string datatype variables.
6  #include <fstream>  // Fstream files.
7  #include <iomanip>   // fixed, setw, setprecision.
8  #include <ostream>
9  #include <sstream>
10 using namespace std;
11
12 enum Menu {
13     EXIT = 0,
14     OutputList,
15     TitleSearch,
16     GenreSearch,
17     ActorSearch,
18     YearSearch,
19     RatingSearch,
20 };
21
22 struct DVDNode
23 {
24     string title;
25     string leadAct;
26     string supAct;
27     string genre;
28     string altGenre;
29     int    year;
30     int    rating;
31     string synop;
32     DVDNode* next;
33 };
34
35 const int ColSpace = 18; // CALC - Column setw space size.
36
37 #endif
38
39
```

```

1  #include "Header.h"
2  #include "Movie.h"
3  #include "Stack.h"
4
5  /
    *****
    **
6  * OPP - DVD Movie List.
7  * -----
8  * This program will keep track of all the Dvds for the user and display them
9  * into an output file.
10 * -----
11 * INPUT:
12 *   userInFile : Input file name.
13 *   userOFile  : Output file name.
14 *****
    /
15 int main()
16 {
17     /
        *****
        **
18     * CONSTANTS
19     * -----
20     * OUTPUT - USED FOR CLASS HEADING
21     * -----
22     * PROGRAMMER : Programmer's Name
23     * CLASS      : Student's Course
24     * SECTION    : Class Days and Times
25     * LAB_NUM    : Lab Number (specific to this lab)
26     * LAB_NAME   : Title of the Lab
27     *****/
28
29     const string AS_NAME = "OPP - DVD Movie List";
30     const int AS_NUM = 5;
31     const string STUDENT_NAME = "Andrew Gharrios";
32     const string CLASS_INFO = "M-Th 5-7:20p";
33     const char AS_TYPE = 'A';
34     const long long STUDENT_ID = 1449366;
35
36     MovieList movie;    // CALC      - Movies list.
37     string userInFile;  // IN & CALC - User input file.
38     string userOFile;   // IN & CALC - User output file.
39
40     cout << left;
41     cout << "*****\n";
42     cout << "*   PROGRAMMED BY : " << STUDENT_NAME << endl;
43     cout << "*   " << setw(14) << "STUDENT ID " << ": " << STUDENT_ID << endl;
44     cout << "*   " << setw(14) << "CLASS " << ": " << CLASS_INFO << endl;

```

```
45     cout << "*" << endl;
46
47     // PROCESSING - This will adjust setw and format appropriately based
48     // on if this is a lab 'L' or assignment
49
50     if (toupper(AS_TYPE) == 'L')
51     {
52         cout << "LAB #" << setw(9);
53     }
54     else
55     {
56         cout << "ASSIGNMENT #" << setw(2);
57     }
58     cout << AS_NUM << ": " << AS_NAME << endl;
59     cout << "*****";
60     cout << right << endl;
61
62
63     cout << "Which input file would you like to use?";
64     getline(cin, userInputFile);
65     cout << "Which output file would you like to use? ";
66     getline(cin, userOutputFile);
67     cout << endl;
68
69     movie.CreateList(userInputFile);
70     movie.DisplayList(userOutputFile);
71
72     return 0;
73 }
74
75
76
77
78
```

```
1  #ifndef STACK_H_
2  #define STACK_H_
3
4  #include "Header.h"
5
6  class StackList
7  {
8  public:
9
10     StackList(); // CONSTRUCTOR
11     ~StackList(); // DECONSTRUCTOR
12
13     /*****
14     **      MUTATORS      **
15     *****/
16     void Push(DVDNode newDVD); // Adds a new Node to the list.
17     DVDNode Pop(); // Removes first node and returns it.
18
19
20     /*****
21     **      ACCESSORS      **
22     *****/
23     bool isEmpty() const; // Returns if list is empty.
24     DVDNode Peek() const; // Returns the first object of the list.
25     int Size() const; // Returns the size of the list.
26
27 protected:
28     DVDNode* head; // CALC - Stack head pointer.
29     int stackCount; // CALC - Amount of objects in stack.
30 };
31
32 #endif
```

```
1  #include "Header.h"
2  #include "Stack.h"
3
4  StackList::StackList() // CONSTRUCTOR
5  {
6      head = NULL;
7      stackCount = 0;
8  }
9
10 StackList::~StackList() // DESTRUCTOR
11 {
12     DVDNode* dvdPtr; // CALC - Dvd pointer.
13
14     dvdPtr = head;
15
16     while (dvdPtr != NULL)
17     {
18         head = head->next;
19         delete dvdPtr;
20         dvdPtr = head;
21     }
22 }
23
24 void StackList::Push(DVDNode newDVD)
25 {
26     DVDNode* dvdPtr; // CALC - Dvd pointer.
27     dvdPtr = new DVDNode;
28
29     *dvdPtr = newDVD;
30
31     dvdPtr->next = head;
32     head = dvdPtr;
33 }
34
35 DVDNode StackList::Pop()
36 {
37     DVDNode* dvdPtr; // CALC - Dvd pointer.
38
39     dvdPtr = head;
40     head = dvdPtr->next;
41     delete dvdPtr;
42     dvdPtr = NULL;
43
44     return *head;
45 }
46
47 bool StackList::isEmpty() const
48 {
49     if (head == NULL)
```

```
50     {
51         return true;
52     }
53     else
54     {
55         return false;
56     }
57 }
58
59 DVDNode StackList::Peek() const
60 {
61     return *head;
62 }
63
64 int StackList::Size() const
65 {
66     DVDNode* ptr; // CALC - DVDpointer.
67     int counter; // CALC - List counter.
68
69     counter = 0;
70
71     while (ptr != NULL)
72     {
73         counter++;
74         ptr = ptr->next;
75     }
76     return counter;
77 }
78
79 /
80 * Push
81 * This method will create a node and add it to the front.
82 * INPUTS:
83 * newDVD : new DVD node.
84 *
85 * No outputs.
86 *
87 *
88 /
89 * Pop
90 * This method will remove the first object and return it.
91 * No inputs.
92 *
```

```
93 * OUTPUTS:
94 *   head : removed Node.
95 *
96 *   *****
97 *   */
98 *
99 *   /
100 *   *****
101 *   **
102 *   IsEmpty
103 *   This method will check if the list is empty and return a boolean
104 *   expression.
105 *   No inputs.
106 *
107 *   OUTPUTS:
108 *   bool : If list is empty or not.
109 *
110 *   *****
111 *   */
112 *
113 *   /
114 *   *****
115 *   **
116 *   Peek
117 *   This method will check the first node of the list and return a copy of it.
118 *   No inputs.
119 *
120 *   OUTPUTS:
121 *   head : First Node of the list.
122 *
123 *   *****
124 *   */
125 *
126 *   /
127 *   *****
128 *   **
129 *   Size
130 *   This method will count the size of the list.
131 *   No inputs.
132 *
133 *   OUTPUTS:
134 *   counter : List size.
135 *
136 *   *****
137 *   */
```

```
1  #ifndef MOVIE_H_
2  #define MOVIE_H_
3
4  #include "Header.h"
5  #include "Stack.h"
6
7  class MovieList : public StackList
8  {
9  public:
10     MovieList(); // DECONSTRUCTOR
11     ~MovieList(); // CONSTRUCTOR
12
13     /*****
14     **      MUTATORS      **
15     *****/
16     void CreateList(string IFileName); // Creates a list of DVDs.
17     void DisplayList(string outputFileName) const; // Displays the list of DVDs.
18
19 private:
20     /*****
21     **      ACCESSORS      **
22     *****/
23     string WordWrap(string plot) const; // returns plot limited to characters per line.
24     string TitleString(string title) const; // returns the Title without exceeding limit.
25     void PrintHeader(ostream& oFile) const; // Outputs the class header to Ofile.
26 };
27
28 #endif
```



```
1  #include "Header.h"
2  #include "Movie.h"
3
4  MovieList::MovieList(){}
5  MovieList::~MovieList(){}
6
7  void MovieList::Createlist(string inFileName)
8  {
9      DVDNode* dvdPtr; // CALC - DVD pointer.
10     dvdPtr = new DVDNode;
11
12     ifstream inFile;
13     inFile.open(inFileName.c_str());
14
15     while (inFile)
16     {
17         getline(inFile, dvdPtr->title);
18         getline(inFile, dvdPtr->leadAct);
19         getline(inFile, dvdPtr->supAct);
20         getline(inFile, dvdPtr->genre);
21         getline(inFile, dvdPtr->altGenre);
22         inFile >> dvdPtr->year;
23         inFile.ignore(10000, '\n');
24         inFile >> dvdPtr->rating;
25         inFile.ignore(10000, '\n');
26         getline(inFile, dvdPtr->synop);
27         inFile.ignore(10000, '\n');
28
29         cout << "Name: " << dvdPtr->title << endl;
30
31         Push(*dvdPtr);
32     }
33     delete dvdPtr;
34     dvdPtr = NULL;
35     inFile.close();
36 }
37
38 void MovieList::DisplayList(string outputFileName) const
39 {
40     DVDNode* dvdPtr; // CALC - DVD pointer
41     int counter; // CALC - Counter
42     ofstream oFile; // CALC & OUT - Output file variable.
43
44     counter = 1;
45     dvdPtr = head;
46
47     oFile.open(outputFileName.c_str());
48
49     PrintHeader(oFile);
```

```

50
51     cout << "\nWriting output file " << outputFileName << endl;
52
53     while (dvdPtr != NULL)
54     {
55         oFile << left;
56         oFile <<
57             "*****" << endl;
58         oFile << "Title: " << TitleString(dvdPtr->title) << endl;
59         oFile << "Title: " << TitleString(dvdPtr->title) << endl;
60         oFile <<
61             "-----" << endl;
62         oFile << "Year: " << setw(ColSpace - 6) << dvdPtr->year << "Rating: " <<
63             << dvdPtr->rating << endl;
64         oFile <<
65             "-----" << endl;
66         oFile << setw(ColSpace) << "Leading Actor: " << setw(ColSpace + 5) <<
67             dvdPtr->leadAct << "Genre 1: ";
68         oFile << dvdPtr->genre << endl;
69         oFile << setw(ColSpace) << "Supporting Actor: " << setw(ColSpace + 5) <<
70             << dvdPtr->supAct << "Genre 2: ";
71         oFile << dvdPtr->altGenre << endl;
72         oFile <<
73             "-----" << endl;
74         oFile << "PLOT:\n";
75         oFile << WordWrap((*dvdPtr).synop);
76         oFile <<
77             "\n*****" << endl;
78             "*****" << endl;
79         oFile << endl << endl;
80         oFile << right;
81         counter++;
82         dvdPtr = dvdPtr->next;
83     }
84     oFile.close();
85 }
86
87 string MovieList::WordWrap(string title) const
88 {
89     const int MAX = 75; // CALC - Maximum characters per line.
90
91     ostream stream; // Stringstream output variable
92     string word; // CALC & OUT - Temp word memory.
93     string line; // CALC & OUT - Temp line memory.
94

```

```
86     word = "";
87     line = "";
88
89     for (unsigned int i = 0; i < title.length(); i++)
90     {
91         if (title[i] != ' ')
92         {
93             word += title[i];
94         }
95         else
96         {
97             if (line.length() + word.length() > MAX)
98             {
99                 stream << line << endl;
100                 line = "";
101             }
102             line += word + " ";
103             word.clear();
104         }
105         if (i == title.length() - 1)
106         {
107             if (line.length() + word.length() > MAX)
108             {
109                 stream << line << endl;
110                 line.clear();
111             }
112             line += word + " ";
113             stream << line;
114             return stream.str();
115         }
116     }
117 }
118
119 string MovieList::TitleString(string title) const
120 {
121     const int MAX = 50; // CALC - Maximum characters per title.
122
123     if (title.length() > MAX)
124     {
125         title = title.substr(0, MAX - 3) + "...";
126     }
127     return title;
128 }
129
130 void MovieList::PrintHeader(ostream& oFile) const
131 {
132     /
133     ****
134     ****
135     ****
136     ****
137     ****
138     ****
139     ****
140     ****
141     ****
142     ****
143     ****
144     ****
145     ****
146     ****
147     ****
148     ****
149     ****
150     ****
151     ****
152     ****
153     ****
154     ****
155     ****
156     ****
157     ****
158     ****
159     ****
160     ****
161     ****
162     ****
163     ****
164     ****
165     ****
166     ****
167     ****
168     ****
169     ****
170     ****
171     ****
172     ****
173     ****
174     ****
175     ****
176     ****
177     ****
178     ****
179     ****
180     ****
181     ****
182     ****
183     ****
184     ****
185     ****
186     ****
187     ****
188     ****
189     ****
190     ****
191     ****
192     ****
193     ****
194     ****
195     ****
196     ****
197     ****
198     ****
199     ****
200     ****
201     ****
202     ****
203     ****
204     ****
205     ****
206     ****
207     ****
208     ****
209     ****
210     ****
211     ****
212     ****
213     ****
214     ****
215     ****
216     ****
217     ****
218     ****
219     ****
220     ****
221     ****
222     ****
223     ****
224     ****
225     ****
226     ****
227     ****
228     ****
229     ****
230     ****
231     ****
232     ****
233     ****
234     ****
235     ****
236     ****
237     ****
238     ****
239     ****
240     ****
241     ****
242     ****
243     ****
244     ****
245     ****
246     ****
247     ****
248     ****
249     ****
250     ****
251     ****
252     ****
253     ****
254     ****
255     ****
256     ****
257     ****
258     ****
259     ****
260     ****
261     ****
262     ****
263     ****
264     ****
265     ****
266     ****
267     ****
268     ****
269     ****
270     ****
271     ****
272     ****
273     ****
274     ****
275     ****
276     ****
277     ****
278     ****
279     ****
280     ****
281     ****
282     ****
283     ****
284     ****
285     ****
286     ****
287     ****
288     ****
289     ****
290     ****
291     ****
292     ****
293     ****
294     ****
295     ****
296     ****
297     ****
298     ****
299     ****
300     ****
301     ****
302     ****
303     ****
304     ****
305     ****
306     ****
307     ****
308     ****
309     ****
310     ****
311     ****
312     ****
313     ****
314     ****
315     ****
316     ****
317     ****
318     ****
319     ****
320     ****
321     ****
322     ****
323     ****
324     ****
325     ****
326     ****
327     ****
328     ****
329     ****
330     ****
331     ****
332     ****
333     ****
334     ****
335     ****
336     ****
337     ****
338     ****
339     ****
340     ****
341     ****
342     ****
343     ****
344     ****
345     ****
346     ****
347     ****
348     ****
349     ****
350     ****
351     ****
352     ****
353     ****
354     ****
355     ****
356     ****
357     ****
358     ****
359     ****
360     ****
361     ****
362     ****
363     ****
364     ****
365     ****
366     ****
367     ****
368     ****
369     ****
370     ****
371     ****
372     ****
373     ****
374     ****
375     ****
376     ****
377     ****
378     ****
379     ****
380     ****
381     ****
382     ****
383     ****
384     ****
385     ****
386     ****
387     ****
388     ****
389     ****
390     ****
391     ****
392     ****
393     ****
394     ****
395     ****
396     ****
397     ****
398     ****
399     ****
400     ****
401     ****
402     ****
403     ****
404     ****
405     ****
406     ****
407     ****
408     ****
409     ****
410     ****
411     ****
412     ****
413     ****
414     ****
415     ****
416     ****
417     ****
418     ****
419     ****
420     ****
421     ****
422     ****
423     ****
424     ****
425     ****
426     ****
427     ****
428     ****
429     ****
430     ****
431     ****
432     ****
433     ****
434     ****
435     ****
436     ****
437     ****
438     ****
439     ****
440     ****
441     ****
442     ****
443     ****
444     ****
445     ****
446     ****
447     ****
448     ****
449     ****
450     ****
451     ****
452     ****
453     ****
454     ****
455     ****
456     ****
457     ****
458     ****
459     ****
460     ****
461     ****
462     ****
463     ****
464     ****
465     ****
466     ****
467     ****
468     ****
469     ****
470     ****
471     ****
472     ****
473     ****
474     ****
475     ****
476     ****
477     ****
478     ****
479     ****
480     ****
481     ****
482     ****
483     ****
484     ****
485     ****
486     ****
487     ****
488     ****
489     ****
490     ****
491     ****
492     ****
493     ****
494     ****
495     ****
496     ****
497     ****
498     ****
499     ****
500     ****
501     ****
502     ****
503     ****
504     ****
505     ****
506     ****
507     ****
508     ****
509     ****
510     ****
511     ****
512     ****
513     ****
514     ****
515     ****
516     ****
517     ****
518     ****
519     ****
520     ****
521     ****
522     ****
523     ****
524     ****
525     ****
526     ****
527     ****
528     ****
529     ****
530     ****
531     ****
532     ****
533     ****
534     ****
535     ****
536     ****
537     ****
538     ****
539     ****
540     ****
541     ****
542     ****
543     ****
544     ****
545     ****
546     ****
547     ****
548     ****
549     ****
550     ****
551     ****
552     ****
553     ****
554     ****
555     ****
556     ****
557     ****
558     ****
559     ****
560     ****
561     ****
562     ****
563     ****
564     ****
565     ****
566     ****
567     ****
568     ****
569     ****
570     ****
571     ****
572     ****
573     ****
574     ****
575     ****
576     ****
577     ****
578     ****
579     ****
580     ****
581     ****
582     ****
583     ****
584     ****
585     ****
586     ****
587     ****
588     ****
589     ****
590     ****
591     ****
592     ****
593     ****
594     ****
595     ****
596     ****
597     ****
598     ****
599     ****
600     ****
601     ****
602     ****
603     ****
604     ****
605     ****
606     ****
607     ****
608     ****
609     ****
610     ****
611     ****
612     ****
613     ****
614     ****
615     ****
616     ****
617     ****
618     ****
619     ****
620     ****
621     ****
622     ****
623     ****
624     ****
625     ****
626     ****
627     ****
628     ****
629     ****
630     ****
631     ****
632     ****
633     ****
634     ****
635     ****
636     ****
637     ****
638     ****
639     ****
640     ****
641     ****
642     ****
643     ****
644     ****
645     ****
646     ****
647     ****
648     ****
649     ****
650     ****
651     ****
652     ****
653     ****
654     ****
655     ****
656     ****
657     ****
658     ****
659     ****
660     ****
661     ****
662     ****
663     ****
664     ****
665     ****
666     ****
667     ****
668     ****
669     ****
670     ****
671     ****
672     ****
673     ****
674     ****
675     ****
676     ****
677     ****
678     ****
679     ****
680     ****
681     ****
682     ****
683     ****
684     ****
685     ****
686     ****
687     ****
688     ****
689     ****
690     ****
691     ****
692     ****
693     ****
694     ****
695     ****
696     ****
697     ****
698     ****
699     ****
700     ****
701     ****
702     ****
703     ****
704     ****
705     ****
706     ****
707     ****
708     ****
709     ****
710     ****
711     ****
712     ****
713     ****
714     ****
715     ****
716     ****
717     ****
718     ****
719     ****
720     ****
721     ****
722     ****
723     ****
724     ****
725     ****
726     ****
727     ****
728     ****
729     ****
730     ****
731     ****
732     ****
733     ****
734     ****
735     ****
736     ****
737     ****
738     ****
739     ****
740     ****
741     ****
742     ****
743     ****
744     ****
745     ****
746     ****
747     ****
748     ****
749     ****
750     ****
751     ****
752     ****
753     ****
754     ****
755     ****
756     ****
757     ****
758     ****
759     ****
760     ****
761     ****
762     ****
763     ****
764     ****
765     ****
766     ****
767     ****
768     ****
769     ****
770     ****
771     ****
772     ****
773     ****
774     ****
775     ****
776     ****
777     ****
778     ****
779     ****
780     ****
781     ****
782     ****
783     ****
784     ****
785     ****
786     ****
787     ****
788     ****
789     ****
790     ****
791     ****
792     ****
793     ****
794     ****
795     ****
796     ****
797     ****
798     ****
799     ****
800     ****
801     ****
802     ****
803     ****
804     ****
805     ****
806     ****
807     ****
808     ****
809     ****
810     ****
811     ****
812     ****
813     ****
814     ****
815     ****
816     ****
817     ****
818     ****
819     ****
820     ****
821     ****
822     ****
823     ****
824     ****
825     ****
826     ****
827     ****
828     ****
829     ****
830     ****
831     ****
832     ****
833     ****
834     ****
835     ****
836     ****
837     ****
838     ****
839     ****
840     ****
841     ****
842     ****
843     ****
844     ****
845     ****
846     ****
847     ****
848     ****
849     ****
850     ****
851     ****
852     ****
853     ****
854     ****
855     ****
856     ****
857     ****
858     ****
859     ****
860     ****
861     ****
862     ****
863     ****
864     ****
865     ****
866     ****
867     ****
868     ****
869     ****
870     ****
871     ****
872     ****
873     ****
874     ****
875     ****
876     ****
877     ****
878     ****
879     ****
880     ****
881     ****
882     ****
883     ****
884     ****
885     ****
886     ****
887     ****
888     ****
889     ****
890     ****
891     ****
892     ****
893     ****
894     ****
895     ****
896     ****
897     ****
898     ****
899     ****
900     ****
901     ****
902     ****
903     ****
904     ****
905     ****
906     ****
907     ****
908     ****
909     ****
910     ****
911     ****
912     ****
913     ****
914     ****
915     ****
916     ****
917     ****
918     ****
919     ****
920     ****
921     ****
922     ****
923     ****
924     ****
925     ****
926     ****
927     ****
928     ****
929     ****
930     ****
931     ****
932     ****
933     ****
934     ****
935     ****
936     ****
937     ****
938     ****
939     ****
940     ****
941     ****
942     ****
943     ****
944     ****
945     ****
946     ****
947     ****
948     ****
949     ****
950     ****
951     ****
952     ****
953     ****
954     ****
955     ****
956     ****
957     ****
958     ****
959     ****
960     ****
961     ****
962     ****
963     ****
964     ****
965     ****
966     ****
967     ****
968     ****
969     ****
970     ****
971     ****
972     ****
973     ****
974     ****
975     ****
976     ****
977     ****
978     ****
979     ****
980     ****
981     ****
982     ****
983     ****
984     ****
985     ****
986     ****
987     ****
988     ****
989     ****
990     ****
991     ****
992     ****
993     ****
994     ****
995     ****
996     ****
997     ****
998     ****
999     ****
1000     ****
```

```

133  * CONSTANTS
134  *
    -----
    -
135  * OUTPUT - USED FOR CLASS HEADING
136  *
    -----
    -
137  * PROGRAMMER : Programmer's Name
138  * CLASS      : Student's Course
139  * SECTION    : Class Days and Times
140  * LAB_NUM    : Lab Number (specific to this lab)
141  * LAB_NAME   : Title of the Lab
142  *****
    /

143
144  const string AS_NAME = "OPP - DVD Movie List";
145  const int AS_NUM = 5;
146  const string STUDENT_NAME = "Andrew Gharrios";
147  const string CLASS_INFO = "M-Th 5-7:20p";
148  const char AS_TYPE = 'A';
149  const long long STUDENT_ID = 1449366;
150
151  oFile << left;
152  oFile << "*****
    \n";
153  oFile << "*" PROGRAMMED BY : " << STUDENT_NAME << endl;
154  oFile << "*" " << setw(14) << "STUDENT ID " << ": " << STUDENT_ID <<
    endl;
155  oFile << "*" " << setw(14) << "CLASS " << ": " << CLASS_INFO << endl;
156  oFile << "*" ";
157
158  // PROCESSING - This will adjust setws and format appropriately based
159  //              on if this is a lab 'L' or assignment
160
161  if (toupper(AS_TYPE) == 'L')
162  {
163      oFile << "LAB #" << setw(9);
164  }
165  else
166  {
167      oFile << "ASSIGNMENT #" << setw(2);
168  }
169  oFile << AS_NUM << ": " << AS_NAME << endl;
170  oFile << "*****";
171  oFile << right << endl;
172 }
173
174 /

```

```
*****
**
175 * CreateList
176 *   This method will create a list for all the DVDs
177 * INPUTS:
178 *   inFileName : name of input file.
179 *
180 * No outputs.
181 *
*****
*/
182
183 /
*****
**
184 * DisplayList
185 *   This method will display the entire list of DVDs to the output file.
186 * INPUTS:
187 *   outFileName : output file name.
188 *
189 * No outputs.
190 *
*****
*/
191
192 /
*****
**
193 * WordWrap
194 *   This method limit the amounts of characters outputted per line.
195 * INPUTS:
196 *   title : Text to word wrap.
197 *
198 * OUPUTS:
199 *   title : Text that is word wrapped.
200 *
*****
*/
201
202 /
*****
**
203 * TitleString
204 *   This method will make sure the title doesn't exceed the maximum amount.
205 * INPUTS:
206 *   title : Text to word wrap.
207 *
208 * OUPUTS:
209 *   title : Text that is word wrapped.
```

```
210 *  
    *****  
    */  
211  
212 /  
    *****  
    **  
213 * PrintHeader  
214 *   This method will output the class header into the outputfile.  
215 * INPUTS:  
216 *   oFile : Output file ostream variable.  
217 *  
218 * No outputs.  
219 *  
    *****  
    */  
220  
221
```