Problem

Code a Bowling Analysis Checker (BAC) program to check and parse the file format for the bowling science (BS), bowling analysis management (BAM) system that will be developed this term.

Discussion

The BAC program must read and process the BAM file format. (See the input section for a discussion of the BAM file contents.) Only required information from the BAM file must be checked and validated. Any missing required data will not be stored internally, but will be reported to the user in a report file, (see the output section below). The BAM file information must be stored in separate arrays of records, decomposed by different record types. The interface for this program will be extremely simple, consisting of a query and answer dialog with the user. This program will later be incorporated into a bowling analysis list, (BAL), system.

Execution

The BAC program should initially present the user with a startup screen displaying the name of the program, a brief 3-4 line explanation of the program and the programmer's name and email address. After the user hits the return key, the startup screen is to be cleared. A brief help screen should appear explaining the program, which is cleared upon the pressing of the return key.

Following the brief help screen the user is prompted for the name of the BAM file. The name of the file is input and opened for reading. By default the program is to assume that the file is located within the current directory. (The user is responsible for entering a path name specification if the file is located in a different directory.) If the file does not exist, the program is to halt with an appropriate error message. If the file does exist, it is read, parsed and checked for format errors. For each entry of the BAM file that is checked and found invalid, the BAC program will include a brief entry in the report file, and echo it to the screen. (See the Output section for the report file description.) After the file has been processed, scored and all output performed, the program must issue a brief termination message before returning to the operating system.

Input

A BAM contains information for a single individual bowler. The BAM file is composed of eight sections: 1. Alley, 2. Event, 3. Tools, 4. Ball, 5. Release, 6. Conditions, 7. Games and 8. Frames. Each of these sections is delimited by a line with an equals symbol in columns one - ten followed by a text section label, (the text section label is not a requirement and thus does not need to be checked for existence), followed by equal signs. The sections are composed of multiple lines with labels and fields in a strict format. The format for a single entry in each section will be described below. However, be aware that each section may have multiple entries which will be delimited by a line of dashes.

Alley section:

The alley section of the BAM file contains information about bowling establishments and types of lanes. See the table below for a synopsis of the alley section field entries.

line	cols	label	contents	cols	type	limits	comment
1	1-5	Adex:	*alley index*	7-11	int	> -1	
1	13-17	Name:	*name of alley*	19-38	string	none	
1	40-44	Addr:	*street address*	46-75	string	none	
1	76-80	City:	*city name*	82-91	string	none	
1	93-95	St:	*state*	97-98	string	2 letters	abbreviation
1	100-103	Zip:	*zip code*	105-109	int	5 digits	(alternate type: string)
1	111-116	Phone:	*phone number*	118-129	string	none	
2	1-3	Co:	*manufacturer*	5-13	string	none	AMF, Brunswick, other
2	15-22	Surface:	*surface type*	24	char	set:	{ 'W', 'S', 'O', 'U' }
2	26-31	PinWt:	*pin weight oz.*	33-34	int	$\geq 2 \leq 10$	wt range 3.2 .. 3.10

Event section:

The event section will contain description data for each bowling league, tourney, etc. in which a bowler participates. The table below contains the event section field entries.

line	cols	label	contents	cols	type	limits	comment
1	1-5	Edex:	*event index*	7-11	int	> -1	
1	13-18	Event:	*type of event*	20	char	set:	{ '5', '4', 'T', 'S' , 'P', 'O' }
1	22-27	Title:	*event name*	29-48	string	none	

Tools section:

The tools section of the BAM file holds information about a bowler's equipment. The next table describes the tools section field entries.

line	cols	label	contents	cols	type	limits	comment
1	1-5	Tdex:	*tool index*	7-11	int	> -1	
1	13-18	Shoes:	*description*	20-39	string	none	
1	41-46	Glove:	*description*	48-67	string	none	
1	69-74	Other:	*description*	76-85	string	none	

Ball section:

The ball section contains data about the bowling balls used by a bowler. See the table below for a synopsis of the ball section field entries.

line	cols	label	contents	cols	type	limits	comment
1	1-5	Bdex:	*ball index*	7-11	int	> -1	
1	13-15	Co:	*company*	17-36	string	none	
1	38-43	Model:	*model name*	45-64	string	none	
1	66-69	Pds:	*pounds*	71-72	int	>0 & < 16	
1	74-76	Oz:	*ounces*	78-79	int	> -1 & < 16	
1	81-84	Own:	*purchase date*	86-93	date	legal date	mm/dd/yy
1	95-100	Rolls:	*# games*	102-106	int	> -1	games bowled with ball
2	1-5	Trac:	*track code*	7	char	set:	{ 'F', 'H', 'L', 'S', 'O' }
2	9-16	Surface:	*composition*	18	char	set:	{ 'H', 'L', 'P', 'R', 'U', 'O' }
2	20-28	Hardness:	*density*	30-31	int	$\geq$ 72	
2	33-40	ThumbWt:	*weight*	42-46	float	$\geq$ 0 & $\leq$ 1.0	
2	48-56	FingerWt:	*weight*	58-62	float	$\geq$ 0 & $\leq$ 1.0	
2	64-69	PosWt:	*weight*	71-75	float	$\geq$ 0 & $\leq$ 1.0	
2	77-82	NegWt:	*weight*	84-88	float	$\geq$ 0 & $\leq$ 1.0	
2	90-95	TopWt:	*weight*	97-101	float	$\geq$ 0 & $\leq$ 3.0	
2	103-108	BotWt:	*weight*	110-114	float	$\geq$ 0 & $\leq$ 3.0	

Release section:

The release section file holds information about a bowler's ball release . The next table describes the release section field entries.

line	cols	label	contents	cols	type	limits	comment
1	1-5	Rdex:	*release index*	7-11	int	> -1	
1	13-20	Release:	*release height*	22	char	set:	{ 'L', 'A', 'M', 'K', 'T', 'H'}
1	24-28	Foul:	*distance from*	30-33	float	≥ 0 & <16	
1	35-40	Slide:	*distance*	42-45	float	≥ 0 & < 16	
1	47-54	Shuffle:	*walk speed*	56	char	set:	{ 'C', 'S', 'M', 'Q', 'F'}
1	58-63	Speed:	*ball speed*	65-69	float	≥ 0 & < 30	
1	71-82	Revolutions:	*ball rev.s*	84-85	int	≥ 0	
1	87-91	Lift:	*'english'*	93	char	set:	{ 'N', 'T', 'L', 'M', 'H', 'X'}
1	95-99	Loft:	*lane impact*	101-104	float	≥ 0 < 30	

Conditions section:

The conditions section of the BAM file contains information about lane conditions. See the table below for a synopsis of the conditions section field entries.

line	cols	label	contents	cols	type	limits	comment
1	1-5	Cdex:	*conditions index*	7-11	int	> -1	
1	13-16	Oil:	*type of oil*	18	char	set:	{ 'S', 'N', 'O', 'U' }
1	20-26	Length:	*lane distance*	28	char	set:	{ 'S', 'M', 'L', 'U' }
1	30-36	Amount:	*oil quantity*	38	char	set:	{ 'H', 'M', 'L', 'D', 'U' }
1	40-45	Block:	*oil pattern*	47	char	set:	{ 'C', 'B', 'R', 'T', 'L', 'U' }

Game section:

The game section holds individual game statistics for a bowler. The table below gives the game section field data.

line	cols	label	contents	cols	type	limits	comment
1	1-5	Gdex:	*game index*	7-11	int	> -1	
1	13-17	Adex:	*alley index*	19-23	int	> -1	establishment game rolled
1	25-29	Edex:	*event index*	31-35	int	> -1	event of game
1	37-41	Tdex:	*tools index*	43-47	int	> -1	equipment used
1	49-53	Cdex:	*conditions index*	55-59	int	> -1	oil conditions
1	61-65	Date:	*date of game*	67-74	date	legal date	mm/dd/yy
1	76-81	Start:	*start time: 16:30*	83-87	time	legal time	hh:mm
1	89-93	Stop:	*stop time: 18:40*	95-99	time	legal time	hh:mm
1	101-105	Temp:	*temperature*	107-111	float		
1	113-121	Humidity:	*percentage*	123-125	int	≥ 0	
1	127-132	Spare:	*spare method*	134	char	set:	{ '2', '3', 'C', 'O'}

Frames section:

The frames section stores information for each ball a bowler rolls in a game. When more than one ball is rolled in a frame, the balls when occur in the frames section in the same order as rolled in the game. See the table below for the frames section field data.

line	cols	label	contents	cols	type	limits	comment
1	1-5	Gdex:	*game index*	7-11	int	> -1	
1	13-18	Lanes:	*lanes: 01-02*	20-24	ints	> 0	
1	26-31	Frame:	*frame number*	33-34	int	> 0 & < 11	
1	36-40	Bdex:	*ball index*	42-46	int	> -1	
1	48-52	Rdex:	*release index*	54-58	int	> -1	
2	1-9	Approach:	*feet location*	11-14	float	> 0 & ≤ 16	distance from foul line
2	16-20	Feet:	*feet horizontal*	22-25	float	> -41 & < 41	board placement of foot
2	27-31	Mark:	*board target*	33-36	float	> 0 & < 40	aiming point on lane
2	38-46	Accuracy:	*target hit/miss*	48-51	float	> -41 & < 40	
2	53-59	PinHit:	*ball impact point*	61-64	float	≥ 0 & ≤ 40	
2	66-71	Score:	*ball score*	73	char	set:	{ '0' … '9', 'X', '/', '-', 'F'}
2	75-80	Leave:	*spare difficulty*	82	char	set:	{ 'S', 'W', 'D' }

Note that while all labels must be present in the BAM file, not all content fields need to have been entered by a bowler. For content which the bowler has chosen not to enter the field must have a default value signifying an empty value. This default value will depend upon the type of the field, as given below :

type:	int	float	string	char	set	time	date
default:	999…999	999…999	"???"	'?'	'?'	99:99	00/00/00

Index fields cannot contain default values. All indexes must be non-negative values. The indexes in the game and frames sections must exist in the other corresponding sections or an error must be reported. (Previous records with errors may cause a cascade of errors to be reported since their indexes will be reported as missing if they are referenced by later records.) Empty numeric fields will contain all 999's in the BAM file. This can represented as INT_MIN and FLT_MIN internally in the BAC array of records.

Assumptions

It may be assumed, for ease of initial implementation, that the maximum number of any section entries will be ≤ 25 each. (This limitation will be removed in later programs.) Assume that all sections of the BAM file exist, are in the order listed above and contain at least one entry. Numeric fields, (ints, floats), may be assumed to not contain non-numeric symbols, (i.e. they do not need to be input as strings, checking the individual symbols and converting to numeric values before checking their ranges), State abbreviations, zip codes and phone numbers may be assumed to be correct, only being checked for existence. It may also be assumed that indexes within a section are unique. Data files will be provided shortly, available from the course Web site: (`http://ei.cs.vt.edu/~cs2574`).

Structure Charts

The initial structure chart design must be submitted by **Wednesday Feb. 3rd**, in a zip archive through the automated acceptor. Submission formats are described on the course Web site. The chart must be embedded in a single MS Word document file named "`schrt1p1.doc`", on 8.5" X 11" formatted pages. The student's name and email address must be listed at the top of the first page of the file. The chart may drawn using MS Draw, or other drawing program of choice, but must be viewable without problem in the Word document. The "`pledge.txt`" file as described on the Web site program submission format must also be in the archive. A final structure chart reflecting design changes and corresponding to the code must be produced and submitted with the executable. The initial chart will be compared against the final chart to determine the quality of the design.

Output

The BAC program will produce only one output file. Sample BAC error reporting is shown below:

```
BAC Error Report: Bowl1.BAM
==================================

     Alley section report
_____________________________________________________________________________
| rec # | line | field name       | error                                    |
|-------+------+------------------+------------------------------------------
|   2   |  3   | Name: label      | mismatched label name: "Nama:"           |
|   2   |  4   | Surface          | invalid character in field: 'X'          |
_____________________________________________________________________________

     Event section report
_____________________________________________________________________________
| rec # | line | field name       | error                                    |
|-------+------+------------------+------------------------------------------
|   3   |  3   | Event            | invalid character in field: 'U'          |
_____________________________________________________________________________

     Tools section report
_____________________________________________________________________________
No errors.
_____________________________________________________________________________

     Ball section report
_____________________________________________________________________________
| rec # | line | field name       | error                                    |
|-------+------+------------------+------------------------------------------
|   3   |  5   | Own date         | invalid date: "02/29/98"                 |
|   5   | 10   | Hardness         | out of range: 77                         |
|____________________________________________________________________________

     Release section report
_____________________________________________________________________________
No errors.
_____________________________________________________________________________

     Conditions section report
_____________________________________________________________________________
| rec # | line | field name       | error                                    |
|-------+------+------------------+------------------------------------------
|   2   |  2   | Oil: label       | mismatched label name: "Oik:"            |
_____________________________________________________________________________

      Game section report
_____________________________________________________________________________
| rec # | line | field name       | error                                    |
|-------+------+------------------+------------------------------------------
|   6   |  6   | Spare            | invalid character in field: '4'          |
_____________________________________________________________________________

      Frames section report
_____________________________________________________________________________
| rec # | line | field name       | error                                    |
|-------+------+------------------+------------------------------------------
|  11   | 21   | Frame            | out of range: 11                         |
_____________________________________________________________________________
```

The file must be echoed to both the screen and to output text file. The BAM report file, will contain error messages for section entries that are missing data or contain invalid data. The file name will be the same as the user entered BAM file, but with a '`.BAC`' extension. Any record/section entry that contains an error need not be stored in the arrays of records with valid entries, (i.e. however, invalid fields may be replaced with default values in order to store the record to eliminate error cascading). For records that might contain multiple errors, BAC need only identify and report the first error in a record, (but checking for and reporting multiple record errors is permissible). Valid dates beyond the current date should not be treated as an error, only the validity of dates need be checked. The record and line number counts listed in the report file should be determined without counting the lines of dashes record delimiters within the sections.

Grading

The due date for this program is Wed., **Feb. 17th**. Arrays of struct, (records), must be used in this program. Separate compilation is also required for this program. Failure to produce a separately compiled program will result in grade penalties. Additionally standard C++ class libraries must be used instead of the non-OOP (Object Oriented Programming) libraries. This requires the `using namespace std`, (standard). (Refer to the CS1044 notes on C++ Strings for examples, (see `http://ei.cs.vt.edu/~cs1044/Notes/Chap12.Strings.pdf`).

Optionally, for those students who have experience implementing C++ classes and dynamic linked lists, an alternative implementation using these features is permitted. Be forewarned, if you have not previously written programs with these constructs you are advised not to try to learn them on the fly and utilize them. To do so and failing to produce a working program will have severe grade penalties and will not constitute a basis for an argument of your grade!

The submitted archive must contain soft copies of the initial structure chart, final structure chart, (in MS Word file .doc format named "`schrt2p1.doc`".), MS Visual C++ IDE project files, (the project workspace file must be named "`BAC.DSW`"), source code, input/output files, executable image (named "`BAC.EXE`"), pledge statement file, and an ASCII readme file, (named "`BAC.TXT`"): with execution instructions, the name of the project, programmer's name and email address, this course number (CS2574), and lecture section (day/time). Protection or encryption must not exist on the archive or any files.

To receive partial credit for programs that are non-working, or are not fully functional, a one or two paragraph description of each of the problem(s) with the program must be included in the assignment archive in a text file named "`PARTIAL1.TXT`"). The location, routine minimum, of each suspected problem/error must be specified along with possible corrections that need to be made.

Future

Note that there are two features that BAC is not required to perform. BAC is not required to score each game record and produce a frame by frame score output file. In addition, BAC is not required to write a 'clean', (omitting any discovered errors), BAM formatted file according to the strict section formats described previously. However, the next version of the system will be required to implement these features and others.