UNLV UNIVERSITY OF NEVADA, LAS VEGAS

Homework Assignment

Class:	CS202	Semester:	Spring 2021
Assignment topic:	Final project	Assignment no.	10

Goal
Combine multiple topics to build one application

Phone address book
With limited resources (fragmented memory) you receive a task to implement address book app based on the linked list. In this assignment, you'll face many practical problems that appear during software development.

The summary of functionalities that you need to implement is as follows:
- load records of the address book from text *csv* file
- store the address book records in a form of doubly linked list
- scroll the list on the screen, displaying three records at the time
- display all the records
- search the list: based on specified criteria and search keyword, display all the matching records, using separate linked list.
- implement hierarchy for the address book records
- use menu to interact with the user
- use exception handling with your own exception class, derived from C++ hierarchy
- use c++11

Hierarchical view
Your address book implements hierarchy – this means that each record can have parent/child. Each record has **child** pointer, that points to the child element. Each record can have no child, or have one child. Record can't have more than one child.

Input file
The data input file is a *csv* file, each line is one record. Fields are as follows:

Field	Meaning
0	record id
1	id of parent record
2	first name
3	last name
4	address: street
5	address: city
6	address: state
7	address: zip code
8	address: mobile number
9	notes

The following structures and classes you need to implement:

Class addrBookExc

Exception class for your application. Derive this class from `runtime_error` and define the constructor with *string* parameter, that will call the constructor of `runtime_error` and pass the parameter value to it.

Class Menu

This is a helper class to implement menu-based interaction with the user. Multiple menu entries can be added, each menu entry is a struct:

menuEntry <<struct>>
+text: string
+key: char

`text` – the text for the menu entry
`key` – a character that is associated with the menu entry

Structure of the *Menu* class:

Menu <<class>>
-title: string
-options: vector<menuEntry>
-validate(char): void
+Menu(string):
+add(string, char): void
+displayAndRead(): char

`title` – the title of the menu, displayed above the options. See the example in the `void search()` function, where the menu with "*SEARCH MENU*" title is shown.

`options` – list of options, populated by `Menu::add()` function. This list is implemented as a vector containing elements of type `menuEntry`.

`void validate(char)` – checks if menu option character is valid, i.e. if it is assigned to any of the options. Menu entries are stored in vector, so you need to iterate through the vector, and:

- if option is found, then exit the function
- if all menu entries were checked, and menu option hasn't been found, then throw `addBookExc` exception with the text "*invalid menu entry*". In the related `catch` block, use `what()` function of exception object and print: "*ERROR:* " followed by `what()` message. Then rethrow the caught exception.

`Menu(string)` – constructor, string parameter is a title. Constructor sets `title` member to the value of the constructor's parameter.

void add(string, char) – adds a **menuEntry** record to the vector, values of **menuEntry** object data members **text**, **key** are set to the **string** and **char** parameters respectively.

char displayAndRead() – function displays the menu: all the menu options stored in the vector, and prompts user for the selection. The character corresponding to the option is validated using **validate()** function. If character is successfully validated – it is converted to lowercase and returned. If validation fails, (for example, for the case when user enters non-existing option) the exception from **validate()** function is caught, message *"try again"* is displayed, and menu is displayed again and user is prompted for menu selection again (this means, that you need to use **try...catch** and validate in **try** block)

Struct *record*

Struct **record** is a data structure to contain data for a single **addressBook** entry (node of the list).

record <<struct>>
+id: int
+first_name: string
+last name: string
+address_street: string
+address_city: string
+address+state: string
+address zip: string
+mobile_no: string
+notes: string
+*child: record
+*prev: record
+*next: record

This struct is used in the class **addressBook** which is the main class of your application.

The structure of the `addressBook` class is as follows:

Class addressBook

Class `addressBook` implements doubly linked list to store all the address book entries, that it reads from the text file.

addressBook <<class>>
-*head: record
-*last: record
-*firstToDisplay: record
-loadFile(string): void
-parseAllocate(string): record*
-addRecord(string): void
-displayRecord(record*): void
-displayAll(record*): void
-displayTree(record*, int): void
+addressBook(string):
+~addressBook():
+display3(): void
+goToFirst(): void
+display(const char): void
+search(): void
+operator++(): void
+operator--(): void

Description of class members:

`*head` – head pointer of the doubly linked list

`*last` – pointer to the last element of the doubly linked list

`*firstToDisplay` – pointer to an element in the list, starting from which the list elements are displayed in the three-record mode (using **`display3()`** function). Unless user wants to scroll the list, the **`*firstToDisplay`** pointer has the same value as **`*head`**. This pointer is a *scroll control* – determines the beginning of displaying in the three-record mode.

`void loadFile(string)` – function to load the input file *records.csv* located in the same directory (hardcoding is allowed). It uses exception handling to handle the case when the input file is missing. If the input file is missing, then the **`addrBookExc`** exception is thrown with the message *"error opening the file: FILENAME*
ABORT"

In the `catch` block the message is printed using **`what()`** function, and the same exception is rethrown. In case the input file exists, it is read line by line, and for each line the **`addRecord()`** function is called.

`record* parseAllocate(string)` – receives a line read from the input file as a parameter. Allocates memory for `record` type and fills in the data fields. Once the object is ready, then it returns the pointer to the allocated object.

`void addRecord(string)` – receives a record in a form of comma separated values (from `loadFile()` function). Calls `parseAllocate()` function to get new linked list element (node), ready to be added. Next step is to update **child** of the record. This is the address of the node, that has id of currently being added record in the *parent* field (field 1 of the line). You need to check if node with id specified as parent (2nd field of the line) is already in the linked list. If yes, set its **child** value to the address of currently added object. Lastly, the newly created node is added at the end of the linked list.

`addressBook(string)` – parametered constructor. Initializes class variables to appropriate values and calls `loadFile()` to populate the linked list.

`~addressBook()` – destructor: deallocates the linked list.
`void displayRecord(record*)` – takes pointer to the `record` type and displays the values in the multi-line format:

```
Name:       Dax Atkinson
Street:     531 Prince Ave.   City: Floral Park         State: NY   ZIP: 11001
Phone:      391-437-0175
Notes:      Met at the AU trip
```

Add newlines when necessary.

`void displayAll(record*)` – displays the entire `addressBook`, starting from the first record. Uses `displayRecord()` to display single record. The code must handle all cases: when list is empty, non-empty etc. This function is recursive, so after displaying a node – it must do recursive call with the address of next one. Note, that this means that the code:

```
record *current = head;
while (current!=nullptr) {
    // display
    current=current->next;
}
```
can NOT be used here.

`void displayTree(record* current, int indent)` – this function displays all the records in the compact form – only *id*, *first name* and *last name* are displayed. The hierarchical structure is presented in this function. Each further level of tree hierarchy is represented by `\t` tab character – so each instance of the `displayTree()` function will start with printing as many `\t` characters as the value of `indent` parameter. See sample output. This function is recursive.

`void display3()` – displays only three records on the screen, using `displayRecord()` function. The address of the first element to be displayed is stored in the `*firstToDisplay` pointer variable. The code of this function must handle all types of situations: when list is empty, list has 1, 2, 3, 4 and more elements, `*firstToDisplay` points to last element (or second to last) etc.

`void goToFirst()` – sets the scrolling control to the beginning of the list and displays three records.

`void display(const char)` – this is a routing function (public) that will do the further call to private function members, either `displayAll()` or `displayTree()`. This is determined by the char parameter:
0=call `displayAll()`
1=call `displayTree()`

`void operator++()` – (prefix) advances the *scroll control* to the next element of the list and displays three records. Must handle all cases – if *scroll control* indicates last record – only one record is displayed (not three), user can't advance beyond the list etc. When *scroll control* is pointing to the last element – calling this function has no visible effect.

`void operator--()` – (prefix) sets the *scroll control* to the element previous to the one that is currently displayed as the first of three records. Must handle all cases – when already pointing to first element, then calling this function has no visible effect.

`void search()` – lists all records that satisfy specified criteria. First, this function creates `Menu` object and adds the following options to the menu:

```
SEARCH MENU
===============
First name   (1)
Last name    (2)
City         (3)
State        (4)
Back to previous menu    (q)
```

This menu is then displayed, and user prompt is invoked. If user selects '*q*' then program goes back to the main menu (see `main.cpp` description). For options 1-4, the `search()` function asks user to enter the *keyword*. Once the keyword is provided, the `search()` function creates an object of `searchList` class, named `searchListObj`. Then, still inside of `search()` function – the address book entries are checked against the search criteria (one of 1-4) and matching records are added to the `searchListObj`, using `searchList` class members. Finally, the search results (i.e. elements of the `searchListObj`) are displayed in the compact form.

Class searchList

This class handles the list of elements of the main list (doubly linked list inside of `addressBook` class), that match the criteria specified by user in the `addressBook::search()` function. This class implements single linked list with the following struct as a node:

searchListEl <<struct>>
+*addrBookElem: record
+*link: searchListEl

Each **`searchListEl`** has a <u>pointer</u> to the *record* type as a <u>data field</u>, and the link to the **`searchListEl`** as a *link/next* field.

The **`searchList`** class implements a single linked list (search list) that will contain all the records that are the result of the search. It has the following structure:

searchList <<class>>
-*head: searchListEl
-*last: searchListEl
+searchList():
+~searchList():
+add(record*): void
+display(): void

`*head` – pointer to the first element of the list
`*last` – pointer to the last element of the list

`searchList()` – constructor: initialize class data members to appropriate values.
`~searchList()` – destructor: deallocate the single linked list

`void add(record*)` – adds element to the search list. Takes the parameter – pointer to the **`record`** type. First it needs to check if the parameter is not a null pointer, if not – further steps are taken. New element of **`searchListEl`** is allocated and the pointer passed as the parameter is stored in the newly created node. Then, this new element is added at the end of the list.

`void display()` – displays all the records, that are referenced by list's nodes. The display is in the condensed, one-line format: implement displaying code inside of **`searchList::display()`**.

```
Choice: 4
Enter keyword/string: FL
searching against the keyword: FL

Records matching:
Aydan     Lyons    7 Del Monte Road    Sarasota           FL  34231  phone=418-843-8331  Met in California
Greyson   Rowland  348 Talbot St.      Longwood           FL  32779  phone=514-974-8811  Met at graduation party
Jakob     Brock    598 Fulton Rd.      Winter Haven       FL  33880  phone=801-416-3589  Insurance agent
Alisson   Walter   65 Logan Dr.        Miami              FL  33125  phone=590-311-4501  Dads family
Gregory   Caldwell 326 W. Wood Ave.    Port Saint Lucie   FL  34952  phone=297-310-0427  Grandpas brother
```

Use **`setw()`** to format the output.

main.cpp

The operation of the main program:

- **`addressBook`** object named **`myBook`** is created: pointer is first declared, then the actual object is dynamically created using **`addressBook`** constructor
- dynamic creation of the **`myBook`** object must be placed in the **`try...catch`** statement. Exception thrown by **`addressBook`** class member function must be also caught in **`main()`**. If exception is caught, then message: *FATAL ERROR: failed to read file, terminating.* is displayed and program is terminated.
- *Menu* object is created with the following options:

```
MAIN MENU
==============
Next                    (s)
Previous                (w)
Go to first             (f)
Display all             (d)
Display tree            (t)
Search                  (j)
Quit                    (q)
```

- first, 3 records are displayed (default view) and menu after that
- depending on the user's selection, appropriate action is taken. Use *switch* statement.
- when user chooses option '*q*' then program exits (performing some actions before that, if necessary).

General remarks

- modify `main.cpp` file only at spots indicated by comments. Do not alter the remaining code – only additions are allowed.
- do not use `using namespace std` in the `.h` files
- Ensure proper memory management and make sure there are no memory leaks
- use `valgrind` tool to confirm the proper memory management. Use the command:

```
valgrind  --tool=memcheck  --leak-check=yes  --show-reachable=yes
--num-callers=20 --track-fds=yes ./main
```

where `main` is the name of tested binary file

Writeup

Answer the following questions in writing. Elaborate on each question, but focus on the point and make your answer as precise as possible.

1. Why `main.cpp` file implements the `myBook` as a pointer and dynamic allocation, instead of creating non-dynamic object?
2. What are the advantages of unordered map?
3. Why it is a good practice to derive own exception class from the C++ exception hierarchy?

How to proceed with this assignment

The structure of this address book application might sound complicated, but when implementing part by part (and testing each part after implementation) it will become easy. Please consider the following steps in your implementation:

1. Implement exception, menu, doubly linked list and loading file along with parsing (without parent/child)
2. Implement all the `addressBook` functions, except parent/child
3. Add functions related to parent/child functionality
4. Implement `searchList` class and its functionality

Test each function after it's implemented, so when moving forward – you can be sure that past code is correct. Proceed step by step, function by function.

Link to the video: https://youtu.be/_dVTbrSSBrM

Provided files

`ast10.pdf`	–	this instruction
`addrBookExc.h`	–	file to define exception class (empty)
`addressBook.cpp`	–	skeleton code for `addressBook` class
`addressBook.h`	–	file to define `addressBook` class
`main.cpp`	–	main program file – add your code, don't alter existing one
`record.h`	–	file to define `record` structure
`searchList.h`	–	skeleton code for `searchList` class
`makefile`	–	make file with compilation command (don't modify)
`records.csv`	–	input data file
`sample_output.txt`	–	sample output of the program

Include the following items in your submission:

No	Element	File
1	exception class	addrBookExc.h
2	definition of *addressBook* class	addressBook.h
3	implementation of *addressBook* class	addressBook.cpp
4	main program file	main.cpp
5	make file	makefile
6	*Menu* class with implementation	menu.h
7	definition of the address book record (entry)	record.h
8	*searchList* class with implementation	searchList.h
9	answers to questions specified in the *Writeup* section	answers.txt
	Summary of the submission	
	Summary: 9 files – zip them and submit the zip file to WebCampus	

See sample output on the next page.

The sample output was generated based on the following input file:

```
1,0,Thalia,Harris,60 S. 6th Dr.,Chillicothe,OH,4560,686-961-6036,Friend from the college
2,1,Dax,Atkinson,531 Prince Ave.,Floral Park,NY,11001,391-437-0175,Met at the AU trip
3,0,Averi,Navarro,7381 Locust Ave.,Monsey,NY,10952,247-925-4306,Moms uncle
4,0,Nico,Fox,69 Mountainview Court,Conway,SC,29526,646-948-1593,Family
5,4,Maci,Rios,8667 Laurel Lane,Lititz,PA,17543,213-462-2748,Family
6,5,Dalia,Gillespie,501 Lancaster Street,New Bern,NC,28560,751-973-5789,From the CS class
7,0,Aydan,Lyons,7 Del Monte Road,Sarasota,FL,34231,418-843-8331,Met in California
8,7,Khloe,Pham,7333 Lilac Dr.,Bear,DE,19701,796-899-6383,Airport friend
9,8,Annabella,Newman,7949 Clay Rd.,Coraopolis,PA,15108,637-506-6720,Moms friend
10,9,Dominic,Keith,302 Forest Rd.,Menomonee Falls,WI,53051,553-426-1242,Taxi driver
11,10,Greyson,Rowland,348 Talbot St.,Longwood,FL,32779,514-974-8811,Met at graduation party
12,11,Emiliano,Carson,9310 Shirley Avenue,Louisville,KY,40207,513-720-0765,Anna's friend
13,0,Jakob,Brock,598 Fulton Rd.,Winter Haven,FL,33880,801-416-3589,Insurance agent
14,13,Cecilia,Browning,604 Edgemont Dr.,Newport News,VA,23601,600-953-5693,Banking advisor
15,14,Nancy,Herrera,608 S. La Sierra Circle,Adrian,MI,49221,876-768-3063,Airline representative
16,0,Emmett,Haynes,256 Vine Street,Beverly,MA,1915,605-726-6745,Work friend
17,16,Alisson,Walter,65 Logan Dr.,Miami,FL,33125,590-331-4501,Dads family
18,17,Gregory,Caldwell,326 N. Wood Ave.,Port Saint Lucie,FL,34952,297-310-0427,Grandpas brother
```

```
YOUR CONTACTS:

Name:        Thalia Harris
Street:      60 S. 6th Dr.     City: Chillicothe State: OH   ZIP: 4560
Phone:       686-961-6036
Notes:       Friend from the college

Name:        Dax Atkinson
Street:      531 Prince Ave.   City: Floral Park State: NY   ZIP: 11001
Phone:       391-437-0175
Notes:       Met at the AU trip

Name:        Averi Navarro
Street:      7381 Locust Ave.  City: Monsey      State: NY   ZIP: 10952
Phone:       247-925-4306
Notes:       Moms uncle


MAIN MENU
==============
Next              (s)
Previous          (w)
Go to first       (f)
Display all       (d)
Display tree      (t)
Search            (j)
Quit              (q)
Choice: w

YOUR CONTACTS:

Name: Dax Atkinson
Street: 531 Prince Ave. City: Floral Park State: NY   ZIP: 11001
Phone:       391-437-0175
Notes:       Met at the AU trip

Name: Averi Navarro
Street: 7381 Locust Ave.      City: Monsey      State: NY   ZIP: 10952
Phone:       247-925-4306
Notes:       Moms uncle

Name: Nico Fox
Street: 69 Mountainview Court City: Conway      State: SC   ZIP: 29526
Phone:       646-948-1593
Notes:       Family


MAIN MENU
==============
Next              (s)
Previous          (w)
Go to first       (f)
Display all       (d)
Display tree      (t)
Search            (j)
Quit              (q)
Choice: d
```

UNLV UNIVERSITY OF NEVADA, LAS VEGAS

Name:	Thalia Harris
Street:	60 S. 6th Dr. City: Chillicothe State: OH ZIP: 4560
Phone:	686-961-6036
Notes:	Friend from the college
Name:	Dax Atkinson
Street:	531 Prince Ave. City: Floral Park State: NY ZIP: 11001
Phone:	391-437-0175
Notes:	Met at the AU trip
Name:	Averi Navarro
Street:	7381 Locust Ave. City: Monsey State: NY ZIP: 10952
Phone:	247-925-4306
Notes:	Moms uncle
Name:	Nico Fox
Street:	69 Mountainview Court City: Conway State: SC ZIP: 29526
Phone:	646-948-1593
Notes:	Family
Name:	Maci Rios
Street:	8667 Laurel Lane City: Lititz State: PA ZIP: 17543
Phone:	213-462-2748
Notes:	Family
Name:	Dalia Gillespie
Street:	501 Lancaster Street City: New Bern State: NC ZIP: 28560
Phone:	751-973-5789
Notes:	From the CS class
Name:	Aydan Lyons
Street:	7 Del Monte Road City: Sarasota State: FL ZIP: 34231
Phone:	418-843-8331
Notes:	Met in California
Name:	Khloe Pham
Street:	7333 Lilac Dr. City: Bear State: DE ZIP: 19701
Phone:	796-899-6383
Notes:	Airport friend
Name:	Annabella Newman
Street:	7949 Clay Rd. City: Coraopolis State: PA ZIP: 15108
Phone:	637-506-6720
Notes:	Moms friend
Name:	Dominic Keith
Street:	382 Forest Rd. City: Menomonee Falls State: WI ZIP: 53051
Phone:	553-426-1242
Notes:	Taxi driver
Name:	Greyson Rowland
Street:	348 Talbot St. City: Longwood State: FL ZIP: 32779
Phone:	514-974-8811
Notes:	Met at graduation party
Name:	Emiliano Carson
Street:	9310 Shirley Avenue City: Louisville State: KY ZIP: 40207

Phone: 513-720-0765
Notes: Anna's friend

Name: Jakob Brock
Street: 598 Fulton Rd. City: Winter Haven State: FL ZIP: 33880
Phone: 801-416-3589
Notes: Insurance agent

Name: Cecilia Browning
Street: 604 Edgemont Dr. City: Newport News State: VA ZIP: 23601
Phone: 600-953-5693
Notes: Banking advisor

Name: Nancy Herrera
Street: 608 S. La Sierra Circle City: Adrian State: MI ZIP: 49221
Phone: 876-768-3063
Notes: Airline representative

Name: Emmett Haynes
Street: 256 Vine Street City: Beverly State: MA ZIP: 1915
Phone: 605-726-6745
Notes: Work friend

Name: Alisson Walter
Street: 65 Logan Dr. City: Miami State: FL ZIP: 33125
Phone: 590-331-4501
Notes: Dads family

Name: Gregory Caldwell
Street: 326 N. Wood Ave. City: Port Saint Lucie State: FL ZIP: 34952
Phone: 297-310-0427
Notes: Grandpas brother

```
MAIN MENU
==============
Next                (s)
Previous            (w)
Go to first         (f)
Display all         (d)
Display tree        (t)
Search              (j)
Quit                (q)
Choice: t

TREE:
id=1  name=Thalia Harris
      id=2  name=Dax Atkinson
id=3  name=Averi Navarro
id=4  name=Nico Fox
      id=5  name=Maci Rios
            id=6  name=Dalia Gillespie
id=7  name=Aydan Lyons
      id=8  name=Khloe Pham
            id=9  name=Annabella Newman
                  id=10 name=Dominic Keith
                        id=11 name=Greyson Rowland
                              id=12 name=Emiliano Carson
id=13 name=Jakob Brock
      id=14 name=Cecilia Browning
            id=15 name=Nancy Herrera
id=16 name=Emmett Haynes
      id=17 name=Alisson Walter
            id=18 name=Gregory Caldwell


MAIN MENU
==============
Next         (s)
Previous     (w)
Go to first  (f)
Display all  (d)
Display tree       (t)
Search             (j)
Quit         (q)
Choice: j
Choose search criteria:


SEARCH MENU
==============
First name   (1)
Last name    (2)
City         (3)
State        (4)
Back to previous menu    (q)
Choice: 4
Enter keyword/string: FL
searching against the keyword: FL
```

```
Records matching:
Aydan     Lyons     7 Del Monte Road     Sarasota          FL    34231  phone=418-843-8331   Met in California
Greyson   Rowland   348 Talbot St.       Longwood          FL    32779  phone=514-974-8811   Met at graduation party
Jakob     Brock     598 Fulton Rd.       Winter Haven      FL    33880  phone=801-416-3589   Insurance agent
Alisson   Walter    65 Logan Dr.         Miami             FL    33125  phone=550-331-4501   Dads family
Gregory   Caldwell  326 N. Wood Ave.     Port Saint Lucie  FL    34952  phone=297-310-0427   Grandpas brother

MAIN MENU
===============
Next             (s)
Previous         (w)
Go to first      (f)
Display all      (d)
Display tree     (t)
Search           (j)
Quit             (q)
Choice: q
```