CSC215 Programming Techniques – Due February 26 at Midnight
Programming Assignment #2: The SRT Editor

Program Statement:

Your instructor wishes to be able to edit srt files. They are the simplest form of file that provides movie subtitles. Your program will allow the user to perform many operations to create, edit and destroy frames within these file types.

Program Startup:

I will start your program with one command line argument. This will be the name of the srt file. If the file does not exist, there is nothing to read and should proceed to the main menu. The file name given on the command prompt will be the default file name for saving the srt list. If the file exists, insert all the frames within the file into your list. I guarantee that the file contents will be a valid SRT file. If no file name or to many file names are given, output an error message, a usage statement and exit the program.

C:\> prog2a.exe starWars1.srt

SRT File Format:

There are four parts to a subtitle, the frame number, the starting time, the ending time and the caption to be displayed. Each subtitle must have all four of these things. A blank line will separate each frame. The frame numbers start at 1 and increment in order. The starting time is the time into the movie that the caption should be displayed. The ending time is the time into the movie that the caption will be removed from the screen. The caption contains the lines to be displayed. Many lines can make up the caption. The end of the caption is found when you hit a blank line. You must preserve multiple line captions.

A sample srt file

```
1
00:00:08,000 --> 00:00:10,000
Nothing is going on.

2
00:00:10,500 --> 00:00:12,500
You liar!

3
00:00:13,500 --> 00:00:15,000
Are you?

4
00:00:17,000 --> 00:00:20,000
Violet, please!
- I am not your friend!

5
00:00:24,000 --> 00:00:29,000
You stupid dog,
look what you gone and done now, ay.

6
00:00:34,000 --> 00:00:36,000
Vi, please.
Leave me alone!

7
00:00:36,000 --> 00:00:38,500
- We need to talk.
- Jason, are you deaf?!

8
00:00:41,000 --> 00:00:43,000
What's going on?
```

The start time in the file is in the format of HH:MM:SS,milli. After the starting time there is 1 space, then 2 dashes and the greater than symbol followed by another space. The last item on the line is the ending time for the frame. It is also in the form of HH:MM:SS,milli. All spaces of both times must be filled. The next n number of lines represent the caption to be displayed on the screen. The caption ends when a blank line is found. This blank line separates one frame from the next. This pattern will repeat till the end of the file.

Program Tasks:

When your program is started, you will create an empty SRT list. Then a menu will be presented to the user for tasks to be performed. You must use the menu described later in this document. I will press integer values only at the menu followed by the <enter key> to input my menu option. If an invalid value is entered, display an error message and continue prompting for a selection. This will continue until 7 is pressed..

Your program will give the user several options for manipulating this data.

Validating the time:

Both the start time and end time will need to be validated when the user inputs this data. A valid time has 4 values. An hour, a minute, a second and the milliseconds.

HH:MM:SS,milli
HH must be 2 digits, ex. 11 or 03
MM must be 2 digits, ex. 37 or 09
SS must be 2 digits, ex. 58 or 06
Milli must be 3 digits 587 or 002
Colons separate the hours and minutes, minutes and seconds
A comma separates the seconds and milliseconds.

You may assume that if a section of time is all digits that is within the range for that time piece.
Example: 01:61:45,600 ← 61 will never be provided because 61 is not valid for minutes.
Example: 01:0t:56,601 ← t is not a valid digit and should produce an error message

The Linked List:

To represent the srt file, you will use a class with a singularly linked list. Each node within the linked list will represent a subtitle frame. You must use the following structure for your linked list class

```
struct node
{
    string startTime;
    string endTime;
    string caption;
    node *next;
};
```

No other fields may be added to this structure.

Your class stores all the movie frames will be organized by position based on the starting and ending times of the frames. Frame 1 will be in the first node, frame 2 in the second node, frame n in the nth node. You will not store the frame number because a traversal will output the frames in order with new numbers if necessary.

Example if I insert a new frame at position 2 in the list, when you output the new file, the frame that was in position two will now contain the frame number 3. Therefore, the frame number is not stored in the structure.

Your class definition will the following. If you need another function you **must visit with your instructor** to explain why you need it. You must show me a valid prototype of the function you would like.

```
class srtList
{
    public:
        srtList();
        ~srtList();

        bool insert ( string startingTime, string endingTime,
                      string theCaption );
        bool remove ( int frame );
        bool retrieve ( int frame, string &startingTime, string &endingTime,
                        string &theCaption );
        bool update ( int frame, string theCaption );
        void printSrt ( ostream &out );
        int size();

    private:
        struct node
        {
            string startTime;
            string endTime;
            string caption;
            node *next;
        };

        int count;  // this is optional
        node *headptr;
};
```

Please note, this list will be ordered by the starting and ending times. Any references to a frame number assume that the first node is position 1.

Variable Definitions:

frame:	what frame number to interact with. Counting starts at 1
startTime & startingTime:	a valid starting time to start displaying the caption: must be in format of "HH:MM:SS,mmm"
endTime & endingTime:	a valid ending time to stop displaying the caption: must be in format of "HH:MM:SS,mmm"
caption & theCaption:	the caption to be displayed on the screen.
ostream:	an output stream. Can be ostream (cout) or an ofstream (fout)
count:	keeps track of the number of frames in the linked list
headptr:	contains the address of the first node in the linked list

SRTLIST insert:
A valid starting and ending time will be passed to this function along with a caption that is not empty. I will traverse the list inserting the new frame into its correct position. A frame's starting time must be greater than the ending time of the previous frame, if one exists. A frame's ending time must be less than the starting time of the next frame, if one exists. This function will return true if it successfully adds a frame to the list and false otherwise.

SRTLIST remove:
A frame number will be passed into this function to be removed. Remember counting starts at 1 not 0. This function will return true if it was able to remove the requested frame or false it was unable to remove it.

SRTLIST retrieve:
A frame number will be passed into this function. If the frame number is invalid, false will be returned. If the frame number is valid, the starting and ending times values will be copied into the time strings passed in and the caption will also be copied to the string variable theCaption. If valid, true will be returned.

SRTLIST update:
A frame number and new caption will be passed into this function. If the frame number is valid, the new caption will be copied to the frame and true will be returned. If the frame is invalid, you will return false.

SRTLIST printSRT:
A valid ostream (cout) or ofstream (fout) will be passed into this function. The list will be copied to this ostream following the srt frame format. You must produce a valid srt output.

SRTLIST size:
Will return the number of frames in the list. If you implement count, you will not have to traverse the list counting nodes, but you will need to increment and decrement accordingly when you insert and remove frames.

The menu:
You will present the following menu to edit the srt list. A menu class (see below) will be implemented. The number of frames will be outputted before calling the menu classes print function to display the menu.

```
Frames: 105
  1) Add a Subtitle Frame
  2) Remove a frame
  3) Retrieve a frame
  4) Update a frame
  5) Print subtitles to screen
  6) Print range of subtitles
  7) Exit
Enter Choice:
```

Insert a Subtitle Frame Option:

You will prompt the user for the starting time, the ending time, and the caption.
The times will always be entered as a string and will have to test the times for a valid format. To enter the caption, you must continue to add lines to your string until I enter a blank line. Make sure you separate each line with a newline character. Once you have the information, you will insert it into the frame list using the starting and ending times to determine the position. You will need to validate that no times overlap or are equal.

After inputting all 3 pieces of information, validate the data. Output error messages as appropriate and continue to the main menu.

- Starting time is not in a valid format.
- Ending time is not in a valid format.
- Starting time must come before ending time.
- The caption is empty

If the data is valid, insert the frame information into the list. If the list can successfully insert it based on the starting and ending times, a true will be returned. If you cannot insert the frame the list, output the following error message and continue to the main menu.

- Unable to insert the frame into the SRT list

Remove Frame Option:

You will prompt for the frame number to be removed. Then remove the appropriate frame from the list. If you are unable to remove it from the list, output an error message and continue to the main menu.

- Unable to remove frame # from the SRT list

Retrieve a Frame Option:

You will prompt the user for a frame number to view. You will then retrieve that frame from your list and neatly display the data to the terminal window. If the data cannot be retrieved from the list, output an error message and continue to the main menu.

- Unable to retrieve frame # from the SRT list

Update Frame Option:

You will prompt the user for a frame number to update. After the frame number has been provided, retrieve the frame and neatly display the data to the terminal window. Then allow the user to enter a new caption until a blank line is entered. After the new caption is provided, update the frame. If the caption is empty, do not do an update and display the main menu. If the frame number requested could not be displayed, output an error message and continue to the main menu.

- Unable to retrieve frame # from the SRT list to be updated.

Print Subtitles to Screen Option:

You will display the list to the screen in a valid srt format instead of a file.

Print Range of Subtitles to Screen Options:

You will prompt for the first and last frame numbers and output just those frames to the screen. If you fail to retrieve a frame, output an error message and try the next one.

- Unable to retrieve frame # from the SRT list

Exit Option:

You will exit the main menu and present a submenu for saving the file. This menu only has two options. Option 1 is to save the file to the file provided by the command line argument argv[1]. Option 2 allows the user to exit without saving to the file. If you would like, Option 3 may be implement to prompt for a different file name an your program will save the srt list to the specified file.

The submenu:

```
  1) Save to sw1.srt and exit
  2) Exit without saving
Enter Choice:
```

The Menu Class

This class will be used to implement both the main menu and the sub menu for when you exit the program. The menu entries will be stored in a vector of strings and will allow a programmer to enter menu entries anywhere within the vector. A programmer may also remove entries from the menu and update a menu entry. Other options include printing the menu, getting a valid choice from the menu, returning the number of entries in the menu and clearing all entries in the menu.

Your class definition will the following. If you need another function you **must visit with your instructor** to explain why you need it. You must show me a valid prototype of the function you would like. I will allow you to add a retrieveMenuItem member function if desired, but you must get the prototype approved.

```
class menu
{
    public:
        menu();
        menu ( vector<string> &menuList );
        menu ( menu &aMenu );

        bool addMenuItem ( string item, int pos );
        bool removeMenuItem ( int pos );
        bool updateMenuItem ( string item, int pos );

        void printMenu();
        int getMenuSelection(bool withMenu = true);
        int size();
        void clear();

    private:
        vector<string> theMenu;
};
```

Please note, this menu position will be based off the counting scheme of 1. The first menu entry in the vector will be stored at position 0 put will be represented by a position of 1.

Variable Definitions:

menuList:	A predefined vector containing the menu entries
aMenu:	a predefined menu class to be copied.
item:	a string representing a menu entry.
position:	a location within the menu items. Positions starts counting by 1.
withMenu:	a bool variable. If true, display the menu before prompting for an input selection.
theMenu:	vector of strings to hold the menu entries.

Menu addMenuItem:
This function will place a menu entry at the given location if the position (starts at 1) provided is valid. If the entry is stored, true will be returned. Otherwise false will be returned.

Menu removeMenuItem:
This function will remove the menu item at the given position (starts at 1). If the item is removed, true will be returned. If it fails, false will be returned.

Menu updateMenuItem:
will update a menu location given by position (starts at 1) with the new entry phrase. If successfully updated, true will be returned. Otherwise false is returned.

Menu printMenu:
Will display the vector of menu entries to the screen nicely formatted.

Menu getMenuSelection:
If a value of true is passed in to the function for withMenu, the menu will be displayed before prompting for any value from the user. If the value is not an entry for the menu, an error message will be displayed, and the function will continue prompting for an integer that represents a valid menu choice. Remember, counting starts at 1. Upon successful data entry, return this value.

Menu size
Returns the number of entries in the menu.

Menu clear
Erases all menu items.

This menu class will be used to set up both the main menus and the submenu in the program.
Examples: you can create the menu in 2 different ways.
Setup the main menu:

```
void setupMainMenu ( menu &ourMenu )
{
    int i;
    vector<string> v = {"Add a Subtitle Frame", "Remove a frame",
                        "Retrieve a frame", "Update a frame",
                        "Print subtitles to screen", "Print range of subtitles",
                        "Exit"
                       };

    for ( i=0; i<v.size(); i++ )
    {
        ourMenu.addMenuItem ( v.at(i), ourMenu.size() + 1 );
    }
}
```

Or set up a menu this way:

```
    vector<string> v = {"Save to " + defaultFilename + " and exit",
                        "Exit without saving"
                       };

    menu subMenu( v );
```

The printMenu example:

Calling mainMenu.printMenu

```
  1) Add a Subtitle Frame
  2) Remove a frame
  3) Retrieve a frame
  4) Update a frame
  5) Print subtitles to screen
  6) Print range of subtitles
  7) Exit
```

The getMenuSelection example

Calling the getMenuSelection with true as a parameter.

```
  1) Save to starwars.srt and exit
  2) Exit without saving
Enter Choice: 0

Invalid option.

  1) Save to sw1.srt and exit
  2) Exit without saving
Enter Choice: 1
```

Calling the getMenuSelection with false as a parameter.

```
Enter Choice: 0

Invalid option.

Enter Choice: 1
```

Both will return a valid option for the menu.

Development strategies:

1) Get the menu class working with your partners before spring break starts. This will allow you to focus on the linked list portion when you return from Spring Break.
2) After the menu class is working, develop the insert and print function with your partners. Develop both at the same time. DO NOT HAVE ONE PARTNER DO THE INSERT. You can learn more about linked lists by working as a group.
3) Reuse functions when appropriate.
4) Split the work up. Decide who will work on what part of the program.
5) Work on one menu item at a time.

Repository:

As a team member, never ever under any circumstances commit any code that does not compile and run correctly. Doing so will stop your team members from working on the project until you get your code fixed. Your team members cannot fix your code and it will create conflicts. Remember to pull any code from the remote server each time you develop to keep your copy up to date.

Hints:

Work on program in small stages
Do not do long programming sessions
Do not spend hours trying to debug one problem, seek help from a friend or myself
Save and compile often
Do not wait till the last day or two to program the assignment.

Always pass your lists and Menus by reference.

Requirements:

1. Multiple files are required prog2.cpp srtlist.h and menu.h are required names.
2. I leave it up to you on organizing your files for each class. You may have other headers.
3. The Menu class may not call other functions that you have written that are not member functions of the menu class.
4. The srtList class may not call other functions that you have written that are not member functions of the srtList.
5. The class definition and the node definition may only be in srt.h
6. You must write the destructor to free all the memory.
7. Error checking is a must.

Algorithm:

a) Process command line arguments
b) If file was provided and it opens, load the file into the list
c) Present menu

```
1) Add a Subtitle Frame
2) Remove a frame
3) Retrieve a frame
4) Update a frame
5) Print subtitles to screen
6) Print range of subtitles
7) Exit
```

d) process the selection
e) Go to c) until and exit (7) is pressed
f) Present exit menu

```
1) Save to sw1.srt and exit
2) Exit without saving
```

g) If requested (1) save the file.

Timeline: (Just a suggestion, you must work on the project each week and show the commits to the repository – EXCEPT SPRING BREAK)

February 26	Get teams assigned. Have one person in the group set up the repository like program 1. Create a project name prog2.sln and add in the required listed in the requirements. I would create multiple files for each team member to avoid conflicts. Examples: For team member Tom tomMenu.cpp, tomsrtList.cpp For team member Ann annMenu.cpp, annsrtList.cpp This way no team member works in another person's file. This avoid conflicts. Make sure and add *.srt and html/ to the bottom of the .gitignore file After it is setup, other team members will need to clone this repository.
February 28	Command line arguments Opening the file if it exists, read and output the data to the screen. This will be replaced by the insert later.
March 2	Have the menu class mostly implemented. Make sure constructors, addMenuItem, print Menu and getMenuSeletion are working. The others can be done later.
March 3-11	**ENJOY SPRING BREAK**
March 16	Menu options 1 & 5 should be completed. Replace output of file data with function call to build the list.
March 19	Menu options 2, 3 & 4 should be completed
March 23	Menu options 6, 7 (options 1 & 2 from exit Menu) should be completed
March 24	Test and debug the first part
March 26	Program is DUE. Make sure it is pushed into the server repository. Fill out team evaluation and turn in. More instructions to follow.