**Assumptions:**

* Program automatically throws away a bad data line when reading from an input file
* User inputs correct data type in correct format
* description is 15 characters or less in size
* For new records assigned date, due date, assignment description, and status will be entered in at same time from the main menu.
* A completion date or status will not be entered in the add assignment at the main menu by the user. The status will automatically be set to “assigned” and inserted in the assigned list. Any completion or late status must be entered using the “Complete assignment” selection at the main menu
* Entries have unique assignment dates. There will not be any duplicate assignment dates.
* Exit will not automatically save data. Data must be saved using Save.
* Date check is not against what current real time date is
* All dates are shown in US date format
* An assignment cannot be moved from the completed list back to the assigned list.
* Due date, description, and status edits:
  + Due dates, descriptions, and status cannot be changed once assignment has been moved to the completed list.
  + Status as read from the file will be a string describing the status and not a date.
  + A completion date will be entered from the main menu by the user and will be a date. The date is then compared to the due date to determine if the status is late or completed. After the comparison status is stored in the record as a string. The status date is not retained.
* “Save” at the main menu overwrites the competed assignment file
* Each list has been maintained so that each list is ordered from earliest to latest due date.
* All data after initial installation from file remains sorted by due date
* Data for each assignment will be displayed in a line format with headers. Format to display assignments is as follows:

**List Name**

Due: mm-dd-yyyy

Description: xxxxxxxxxxxxxxx

Assigned: mm-dd-yyyy

Status: xxxxxxxx

**References**

* Phone Book program
  + main.cpp
  + vector\_Based\_Phone\_Dirctory.h
  + User\_Interface.h

**Program credits**

* Source code from Dr Kuhail or Objects, Abstraction, Data Structures, and Design Using C++
  + Date.h
  + const\_list\_iterator.hlist.h
  + list\_iterator.h
  + StringTokenizer.h

**Program Specs**

1. Data stored in the assignment is structured as follows:
   1. The information per assignment record is stored all in one line. Each data value is separated from the next by a comma then a space.
   2. The order is of data values in a record is dueDate, description, assignedDate, status
   3. Example 2-11-2015, linked lists, 2-2-2015, late
2. The program will function as follows:
   1. When the program starts up the following:
      1. two doubly-linked lists with iterators will be generated
         1. Assigned assignments list
            1. Consists of all assignments that have been assigned
         2. Completed assignments list
            1. Consists of all assignments that are:

Completed

Late

* + - 1. A variable will be generated that tracts that data modification has occurred will be set:
      2. Variable set to:
         1. True

Program initialized

Whenever Save is used or Exit is confirmed.

* + - * 1. False

Valid assignment data is read from a file

Whenever assignment data is entered or edited.

* 1. The information furnished when the assignment is first entered will be as follows:
     1. assignedDate – date assignment was assigned, date is in US format
     2. dueDate - date assignment is due, date is in US format
     3. description – description of assignment, description is in string format 15 chars in length
     4. status – enum type that stores as a string the following one of the following status states:
        1. assigned
        2. completed
        3. late
  2. The assignments in both lists will be sorted and stored by due date in the following order: year number, month number and day number.
  3. **Initial assignments will be entered by using a file**. After file has been loaded then all other entries are by user via console menu.
     1. On startup the program will:
        1. Check if the data file name is valid
        2. Open the file
           1. Error message to console if file does not open.
        3. Set variable tracking changes to false
        4. Start reading data lines
        5. Program will check to make sure:
           1. The assignment date is valid

date has been provided

date is not duplicated.

Is a valid month, day, or year value

date is in US format

* + - * 1. description is valid

description is a string of length 15 or less.

* + - * 1. The due date is valid

date has been provided

Is a valid month, day, or year value

date is in US format

due date is not less than assigned date

* + - * 1. status is valid.

Note that status as read from the file is a string and not a date. The sting will be items listed below.

is an string indicating the status is

assigned

completed

late

* + - 1. If valid the program will create a node for each assignment date and assign the node to the correct list based on assignment status and placed in the list in order based on assignment date.
      2. If not valid the program will:

print to console an error message that data line is bad.

automatically throw bad line away

continue reading assignment data lines

* 1. If the initial data input from the file is valid then the program will prompt the user enter a number for what action they would like to do
     1. The following text will print to the console to for this choice:

“Select from the following options:”

“1. Display assignments”

”2. Add assignment”

“3. Edit a due date”

“4. Edit a description”

“5. Complete an assignment”

“6. Display number of late assignments”

”7. Save”

“8. Exit”

“Your input:”

* 1. Based on what the user entered the following will happen:
     1. If the choice was **1. Display assignments**
        1. A list will be printed to the console that consists of both the assignment list and the completed list. The criteria and order for printing is as follows:
           1. Each list will be printed separately to the console and will be ordered by from earliest to latest due date.
           2. The program will first print the assigned list and then the completed list.
           3. The print format for each record printed to console will be as follows:

**List Name**

Due: mm-dd-yyyy

Description: xxxxxxxxxxxxxxx

Assigned: mm-dd-yyyy

Status: xxxxxxxx

* + 1. If the choice was **2. Add assignment**
       1. Console prompts User to enter the assignment date:
       2. Verify that the assigned date is valid as follows:
          1. Date is not a valid date if

Is not US format

Is not a valid month, day, or year value

Assigned date already exists in either list

Note: program will not check if the entered date is valid against a real time date

* + - 1. The program will search both the assigned and completed assignments lists to see if there is a duplicate assignment date
         1. If there is a duplicate assignment date then the program will not change date but will:

issue a warning to the console “Assign Date Error. Date entered has been used previously”.

return to main menu.

* + - * 1. If there is not a duplicate assignment date,

the program will:

prompt the user to enter in the due date

Verify if due date is not a valid date if:

Is not US format

Is not a valid month, day, or year value

Due date is less than the assigned date

Note: program will not check if the entered date is valid against a real time date

If the due date is valid the program will store the due date in variable newAssignDueDate

prompt the user to enter in the description.

verify if the description is valid

15 characters or less.

If the description is valid the program will store the description in variable describe.

If the description is not valid then the program will:

Print error message to console

“Error in description. Returning to main menu”

Returns to main menu

If the description is not valid then the program will:

Print error message to console

“Error in due date. Exiting to main menu”

Returns to the main menu

* + - * 1. if there is Not a duplicate assignment date and the rest of the data entered is valid then the program will:

create a double linked node

set variable to track if there are changes to save to true.

create a new string consisting of existing values formatted as follows.

The information categories will be separated by commas and spaces

dueDate, description, assignedDate, status

status will be automatically set to “assigned”

save the new string in the new node as data

insert node in assigned list in ascending order based on the due date.

returns to main menu once new node has been added to a list.

* + 1. If the choice was **3.** **Edit a due date**
       1. Program requires due date to be in US date format
       2. Assignments that are in the completed list cannot have their data changed.
       3. Program prompts User for assigned date of assignment to be edited.
          1. Runs assignment dates checks as list in ii.
       4. Program then searches the assigned and competed lists for assignment date to see if assignment exists
          1. If assignment does not exist an error message is generated and printed to console.

Program exists to main menu

* + - * 1. If assignment does exist then program

brings back iterator location information for the node

if assignment status is completed then the program:

prints error to console

returns to main menu.

* + - * 1. If assignment is in assigned list then the program:

allows the due date to be change

Prompts the user for a new due date and:

Verifies if due date is an invalid date if:

Is not US format

Is not a valid month, day, or year value

Due date is less than the assigned date

Note: program will not check if the entered date is valid against a real time date

If due date is an invalid date then the program will:

Print error message to console

“Error in due date. Exiting to main menu”

Returns to the main menu

If due date is valid date and the status is assigned then the program will:

Modifies the due date section of the node’s current data string with the new description

Saves the modified string to the data string in the node

Changes the position of the assignment in the list so that the list retains its sort by due date order.

Sets changed variable to true.

Returns to main menu

* + 1. If the choice was **4. Edit a description**
       1. Program prompts User for assigned date of assignment to be edited.
          1. Runs assignment dates checks as list in ii.
       2. Assignments that are in the completed list cannot have their data changed.
       3. Program then searches both lists for assignment date to see if assignment exists
          1. If assignment does not exist an error message is generated.

Print out to console - “Assignment date is not valid. Returning to main menu”

Program exists to main menu

* + - * 1. If assignment does exist then program

brings back iterator location information for the node

If the assignment is in the completed list then the program:

Prints to console “Assignment is completed. Description cannot be changed. Exiting to main menu.”

Returns to main menu

If the assignment status is in the assigned list then the program:

Prompts user to enter new description

Verifies if string length is 15 characters or less

If string length is too long then the program

Prints error message to console

Returns to main menu

If the string is <= to 15 characters then the program:

Modifies the description section of the nodes current data string with the new description

Saves the modified string to the data string in the node

Sets changed variable to true

Returns to main menu

* + 1. If the choice was **5. Complete an assignment**
       1. Program prompts User for assigned date of assignment to be edited.
          1. Runs assignment dates checks as list in ii.
       2. Program then searches assignment list for assignment date to see if assignment exists
          1. If assignment does not exist then the program:

Prints error message to console.

Returns to main menu

* + - * 1. If assignment date record exists and the node is in the completed list then program:

Prints error message to console

Returns to main menu

* + - * 1. If the assignments exist and the node is in the assigned list then the program:

brings back iterator location information for the node

Prompts user to enter a date of when the assignments was completed

The program verifies if the completed date is not valid.

Is not US format

Is not a valid month, day, or year value

Completed date is less than the assigned date

If the completed date is an invalid date then the program will:

Print error message to console

Returns to the main menu

If the completed date is valid then the program

compares completedDate against dueDate

if completedDate <= to dueDate then the program:

modifies status portion of the string data stored in the node as “completed”

if completedDate > to dueDate then the program:

modifies status portion of the string data stored in the node as “late”

saves the modified string to the node

moves the node to the completed list in order of it’s due date

deletes the assignment out of the assigned list

sets the changed variable to true.

returns to the main menu

* + 1. If the choice was **6. Display number of late assignments**
       1. The Program
          1. prompts user to save all changes
          2. searches the completed list for “late” assignment status & counts them
          3. Prints the following output to the console xxx is the number of late assignments

“Number of late assignments is xxxx”

* + - * 1. Returns to main menu
    1. If the choice was **7. Save**
       1. Saves the changes to the data to a file named AssignmentData.txt
       2. Save overwrites the assignment file whenever there are changes to the data.
       3. Sets changed variable to false
       4. Returns to main menu once Save has been executed
    2. If the choice was **7. Exit**
       1. Program will start exiting
       2. the program will check to see if the data has been saved since the last change.
          1. If beenSaved is False then the program will:

Print to console “Data has not been saved. Returning to main menu”

return user to main menu

* + - * 1. If beenSaved is True then the program will

Print to console “Now exiting program”

Exit.

* + 1. If the choice was invalid (not a number that is 1 thru 8) then the program:
       1. prints to console:
          1. “Number entered is not 1 to 8. Please try again.”
          2. Returns to main menu