CS163 Test Plan

Linear_Linked_List

Develop the test plan: *For each member function that you plan to write, think about how to test it – what flow of control exists in the member function and how would you test out all conditions:*

Test Case(s)	Expected Result	Verified? (yes/ no)
push_front on an empty list	Node is added to the front, head AND tail point to the new node	**Yes**
push_front on a populated list	Node is added to the front, head points to the new node	**Yes**
push_back on an empty list	Calls push_front	**Yes**
push_back on a populated list	Node is added to the back, tail points to the new node	**Yes**
add_unique on an empty list	Calls push_front	**Yes**
add_unique on a populated list with a unique item	Item is added to the back of the list	**Yes**
add_unique on a populated list with a duplicate item	Item is not added, function returns false	**Yes**
Calling begin() on an empty list	Begin iterator is equal to end iterator	**Yes**
Calling begin() on a populated list	Begin returns an iterator to the first element	**Yes**
Calling end()	End returns an iterator to a null pointer	**Yes**
Calling size() on an empty list	Returns 0	**Yes**
Calling size() on an N populated list	Returns N	**Yes**
Calling size() after adding an item to an N populated list	Returns N + 1	**Yes**
Calling size() after removing an iterm from an N populated list	Returns N – 1	**Yes**
Calling size() after using the clear() function	Returns 0	**Yes**
Calling empty on an empty list	Returns true	**Yes**

Calling empty on a populated list	Returns false	**Yes**
Calling clear on a populated list	Empty() returns true	**Yes**
Using remove_if with a predicate function	Removes the first data member that returns true from the predicate funciton	**Yes**
Using sort on an unordered list with an ascending comparison function	The list is sorted in ascending order	**Yes**

Verify correctness: Using the above test plan, create a test program that tests the interactions of all functions together.

CS163 Test Plan

Develop the test plan: *For each member function that you plan to write, think about how to test it – what flow of control exists in the member function and how would you test out all conditions:*

Test Case(s)	Expected Result	Verified? (yes/ no)
Project ADT		
Calling name() on a Project object	Returns a read-only reference to the string containing the name	**Yes**
if the Project is default Constructed	The string is an empty string “”	**Yes**
If the Project is value Constructed	The string matches the passed in value	**Yes**
Projects are Ordered by name:		
If Project1 name is “linear lists” and Project2 name is “Binary Trees”		
Then using the ‘<’ operator	Project2 is less than Project1	**Yes**
Using the << operator to display Projects	Project is displayed in a formatted order	**Yes**
Category ADT		
Calling name() on a Category object	Returns a read-only reference to the string containing the name	**Yes**
If the Category is default Constructed	The string is an empty string””	**Yes**
If the Category is value Constructed	The string matches the passed in value	**Yes**
Categories are Ordered by name: If Category1 name is “Assignments” and Category2 name is “Readings” Then using the ‘<’ operator	Category1 is less then Category2	**Yes**
Using add_project to add Project objects		
If the project has a unique name	The project added to the list	**Yes**
If the project is duplicated	The project is not added to the list	**Yes**
Using remove project to delete a project		
If a project matches the target name	The project is deleted	**Yes**
If no projects matches the target	The list is unchanged	**Yes**
Using display_projects on an empty	Nothing is displayed	**Yes**

category		
Using display_projects on a populated list	Each project is displayed	**Yes**

Verify correctness: Using the above test plan, create a test program that tests the interactions of all functions together.

CS163 Test Plan

Develop the test plan: *For each member function that you plan to write, think about how to test it – what flow of control exists in the member function and how would you test out all conditions:*

Test Case(s)	Expected Result	Verified? (yes/ no)
Database ADT		
add_category on an empty or populated database	Adds a category to the collection, initializes it to the name parameter	**Yes**
remove_category on an empty or populated database	Removes a category from the collection.	**Yes**
Using display_categories on an empty database	Displays nothing	**Yes**
Using display_categories on a populated database	Displays each category in the collection	**Yes**
Using add_project		
If the projects category exists	Returns the result of the categories add project call	**Yes**
If the projects category does not exist	Returns false	**Yes**
Using remove_project		
If the projects category exists	Returns the result of the categories remove_project call	**Yes**
If the projects category does not exist	Returns false	**Yes**

Verify correctness: Using the above test plan, create a test program that tests the interactions of all functions together.