Class Names:

We are implementing two classes called GradeBook and ItemType.

DESCRIPTION OF FUNCTIONS:

GradeBook Class Functions:

The class GradeBook will have the following functions:

1. **bool IsFull() const;**
   This function checks whether or not the node is full.

2. **int GetLength() const;**
   This function returns the length.

3. **void MakeEmpty();**
   This function empties out the GradeBook and sets the length to 0.

4. **ItemType GetItem(ItemType& item, bool& found);**
   This function returns the item if it is in the list.

5. **void DeleteItem(ItemType item);**
   This function deletes a location in the node.

6. **void ResetList();**
   This function resets the list by setting the position back to 0.

7. **ItemType GetNextItem();**
   This function returns the item that is next.

8. **void createStudent(ItemType item);**
   This function creates a new student and adds it to the node.

9. **void recordAssignment(int number);**
   This function records specific assignments, it allows the user to input the assignment grade.
10. **void modifyStudent(ItemType& item, bool& found);**
    This function modifies the data of the student.

11. **void recordTest(int number);**
    This function records the test for the student.

12. **void recordFinalExam(int number);**
    This function records the final exam for the student.

13. **void changeGrade(int id, int grade, char choice)**
    This function changes the overall grade for the student.

14. **void calculateFinalGrade()**
    This function calculates the final grade of the student.

15. **void printGradeBook(ofstream& dataFile, GradeBook& GradeBook1)**
    Prints the contents of the GradeBook.

ItemType Class Functions:

The class ItemType will have the following functions:

1. **RelationType ComparedToName(ItemType) const;**
   This function compares the last and first name with another item

2. **RelationType ComparedToID(ItemType) const;**
   This function compares the student ID with another item

3. **void InitializeSemester(int numOfAssignments, int numOfTests, int numOfFinalExam, int assignmentWeight, int testWeight, int finalExamWeight);**
   This function initializes the number of assignments, number of tests, number of final exams, assignment weight, test weight, and the final exam weight for the semester.

4. **void InitializeStudent(int number, std::string firstName, std::string lastName, int id);**
   The function initializes the value, the first name, the last name, and the student ID.

5. **void InitializeGrades(int numOfAssignments, int numOfTests, int numOfFinalExams);**
   The function resizes the assignment, test, and final exam vectors.

6. **void InitializeAssignments(int assignmentNumber, int grade);**
   This function initializes the assignment grades.

7. **void InitializeTests(int tests[]);**
   This function initializes the test grades.

8. **void InitializeFinalExams(int finalExams[]);**
   This function initializes the final exam test grade.

9. **void InitializeFinalGrades();**
   This function initializes the final grades.

10. v**oid Print(std::ostream& out) const;**
    This function prints student's information and grades

FUNCTION TEST CASES:

Test cases for GradeBook Functions:

1. IsFull() has **two** possible test case(s):
   a. The list is full.
   b. The list is not full.
2. GetLength() has **two** possible test case(s):
   a. The length of the list will be returned.
   b. The length of the list will not be returned.
3. MakeEmpty() has **two** possible test case(s):
   a. The list will be emptied.
   b. The list will not be emptied.
4. GetItem(ItemType& item, bool& found) has **two** possible test case(s):
   a. The item is found.
   b. The item is not found.
5. DeleteItem has **two** possible test case(s):
   a. The item will be deleted.
   b. The item will not be deleted.
6. ResetList() has **two** possible test case(s):
   a. The list will be reset and the position will be back to 0.
   b. The list will not be reset and the position will not be back to 0.
7. GetNextTime() has **two** possible test case(s):

a. You get the next item on the list.
b. You will not get the next item on the list.

8. createStudent() has **two** possible test case(s):
   a. A new student is created and added to the list.
   b. A new student is not created or added to the list.
9. recordAssignment(int number) has **two** possible test case(s):
   a. Assignments will be inputted and accurately recorded into the lists and vectors.
   b. Assignments will not be inputted correctly or recorded into the lists and vectors properly.
10. modifyStudent(ItemType& item, bool& found) has **two** possible test case(s):
    a. The data in student will be modified.
    b. The data in student will not be modified.
11. recordTest(int number) has **two** possible test case(s):
    a. The test for student will be recorded.
    b. The test for student will not be recorded.
12. recordFinalExam(int number) has **two** possible test case(s):
    a. The final exam for student will be recorded.
    b. The final exam for student will not be recorded.
13. changeGrade(int id, int grade, char choice) has **two** possible test case(s):
    a. The grade for the student will be changed.
    b. The grade for the student will not be changed.
14. calculateFinalGrade() has **two** possible test case(s):
    a. The final grade is calculated for the student.
    b. The final grade is not calculated for the student.
15. printGradeBook(ofstream& dataFile, GradeBook& GradeBook1) has **two** possible test case(s):
    a. The contents of the GradeBook will be printed.
    b. The contents of the GradeBook will not be printed.

Test cases for ItemType functions:

1. ComparedToName(ItemType) const has **two** possible test case(s):
   a. The value returned will be LESS
   b. The value returned will be GREATER
   c. The value returned will be EQUAL

2. ComparedToID(ItemType) const has **two** possible test case(s):
   a. The value returned will be LESS
   b. The value returned will be GREATER
   c. The value returned will be EQUAL

3. InitializeSemester (int numOfAssignments, int numOfTests, int numOfFinalExam, int assignmentWeight, int testWeight, int finalExamWeight) has **two** possible test case(s):

a. All values have been initialized accurately.
b. One or more values have not been initialized accurately.

4. InitializeStudent (int number, std::string firstName, std::string lastName, int id) has **two** possible test case(s):

a. All values have been initialized accurately.
b. One or more values have not been initialized accurately.

5. InitializeGrades(int numOfAssignments, int numOfTests, int numOfFinalExams) has **two** possible test case(s):

a. All values have been initialized accurately.
b. One or more values have not been initialized accurately.

6. InitializeAssignments(int assignmentNumber, int grade) has **two** possible test case(s):

a. Values for assignments have been initialized accurately.
b. Values for assignments have not been initialized accurately.

7. InitializeTests(int tests[]) has **two** possible test case(s):

a. Values for tests have been initialized accurately.
b. Values for tests have not been initialized accurately.
c.

8. InitializeFinalExams(int finalExams[]) has **two** possible test case(s):

a. Values for final exams have been initialized accurately.
b. Values for final exams have not been initialized accurately.

9. InitializeFinalGrades() has **two** possible test case(s):

a. The final grades for the student will be initialized.
b. The final grades for the student will not be initialized.

10. Print (std :: ostream&) const has **two** possible test case(s):

a. The value of the ItemType will be printed to the stream out.
b. The value of the ItemType will not be printed to the stream out.