Final Project: Journey Through the Genome

Learning Goals

This project will get you more familiar with:

1. File handling and data loading
2. Class design and game architecture
3. DNA sequence analysis in code
4. Creative narrative design for science-based gameplay

Warning: For this project, you are not allowed to use global variables, pointers, pass-by reference, "this->", const reference parameters (e.g., const string &), and range-based for loops (e.g., for(a : b)).

1 Introduction

In this project, you'll design a two-player text-based game involving DNA sequencing of lion conservation genomics. Players act as young genomic scientists competing to become the

next Lead Genomicist on a high-impact conservation team. The project emphasizes design, creativity, genetic reasoning, and software engineering. Along the way, the players will make strategic decisions, face unexpected challenges, and collect Discover Points as they grow and refine other traits, such as accuracy, efficiency, and insight. Whether navigating new samples or working alongside co-workers, each choice will shape their journey. The player who earns the most Discover Points by demonstrating they have what it takes to be the future scientist of DNA sequencing with conservation animals will be chosen as the next Lead Genomicist!

This project has no answer key, prebuilt test cases, or set questions. Instead, you'll be given a list of requirements to meet, and it's up to you to fulfill them in the most creative way possible. This is worth 10% of your final grade.

2 Game Play

The goal of the game is simple: as a player, you'll lead your scientist on one of two distinct paths. Each path offers its own unique set of advantages and challenges, shaping your journey and influencing your growth along the way. As you advance, you'll come across unexpected obstacles that may either help or hinder your progress. Navigate these challenges wisely to prove that you have what it takes to become the next Lead Genomicist.

We have outlined the critical components of the game below, but there are numerous opportunities for creative expression in how you choose to define the details of your game. There is no CodeRunner that you have to match precisely, so take advantage of the flexibility. Whenever we do not explicitly provide details on a given mechanic in the game, please use your own intuition to design an interesting game.

2.1 Starting the Game

This is a two-player game. First, display all available characters and their stats. Then, prompt the players to enter their names and choose a scientist character. Both players choose a character from the predefined list of five scientists (See 2.2). Each character has various personal attributes.

After selecting a character and before the first turn, players also choose their Path Type. This is either Training Fellowships where users take a slower path but develop skills earlier, or the Direct Lab Assignment which is a faster but riskier (See 2.3). Each path type will contain the same number of special and regular tiles. However, the arrangement of the tiles will differ depending on the path chosen (See 3.2).

For each turn in the game, you should display the Main Menu, including the current position of players on the board and relevant options for the space the player is currently on (See 3.1).

2.2 Character Selection Menu

The `characters.txt` file contains a list of playable scientists that players can choose from. Each scientist includes the following attributes: Experience, Accuracy, Efficiency, Insight, and Discovery Points. Ensure that each character can only be chosen once and update the menu to show only the available characters.

Each player begins with 20,000 Discover Points before choosing their Path Type. The Path Type chosen can increase or decrease the number of Discover Points from the characters' starting values. It's important to note that the values for Accuracy, Efficiency, and Insight cannot go below 100 points throughout the entire game. If the values are below 100, default them to 100.

2.3 Path Types

During game setup, ask each player to choose one of two path types. These path types will have an impact on the starting stats for the player as well as the arrangement of the tiles they will follow on the game board. The two path options are listed below:

Training Fellowship: This path equips your scientist with essential traits (accuracy, efficiency, and insight) needed for future challenges. The training requires an investment of -5,000 Discover Points from the starting number of Discover Points. This symbolizes the time and resources dedicated to developing these skills. This path also adds 500 Accuracy Points, 500 Efficiency Points, and 1,000 Insight Points to the starting amount of your character's traits before you start the journey.

- Advisor Choice: If the player selects Training Fellowship, they will be prompted to choose an advisor who grants a unique special ability that protects them during random events that have a negative influence on their Discover Points (See 2.4).

Direct Lab Assignment: This option lets your scientist jump directly into the life of DNA sequencing with an immediate boost of +5,000 Discover Points from the starting number of Discover Points, allowing early progression and quick success. This path also adds 200 Accuracy Points, 200 Efficiency Points, and 200 Insight Points before you start the journey.

Although this path offers a strong head start, it lacks long-term resilience and special abilities that could be gained through mentorship in Training Fellowship, making it a riskier

approach to become a Lead Genomicist. Also, you will not get an initial advisor if you choose this path.

2.4 Advisor List

The advisors have special abilities that can protect you in the case of a negative random event. You should choose your advisor wisely.

1. Dr. Aliquot – A master of the "wet lab", assisting in avoiding contamination.
2. Dr. Assembler – An expert who helps improve efficiency and streamlines pipelines.
3. Dr. Pop-Gen – A genetics specialist with insight for identifying rare genetic variants.
4. Dr. Bio-Script – The genius behind the data analysis, helps debug code.
5. Dr. Loci – Your biggest supporter assisting you in learning the equipment

For example, players on either path could encounter, "A mysterious contamination shows up in your control sample", but if you have Dr. Aliquot as your advisor, you will safely bypass that event. If you do not have Dr. Aliquot as your advisor, you will suffer -300 Discover Points.

2.5 Core Concepts of the Game

1. The game starts by selecting your preferred scientist character from a list of available character names.
2. Select a path for your chosen scientist from the two path types.
3. The map is a two-lane path (one lane for each player). Depending on the path chosen, both players can be on the same path or on different paths. Players navigate by spinning a virtual spinner from 1 to 6 to determine how far to move forward. This can be done using a random number generator.
4. Players alternate taking turns playing each round of the game.
5. There are special tiles on the board game where unique events happen that can affect your Traits.
6. The players must manage their total Accuracy, Efficiency, Wisdom, and Discover Points throughout the game.
7. Negative events occasionally occur, causing players to lose Trait or Discover Points that can setback their journey to become the Lead Genomicist.
8. Once all players reach the Genome Conference, represented by the Orange Tile at the finish line, the player who has the highest Discover Points wins the game.

3 Feature Requirements

3.1 Interactive Components

- Players move forward on the board by spinning a virtual spinner. The number they land on determines how many tiles they advance on their turn. You can do this by using a random number generator.
- Design the two distinct starting paths for players to choose from, Fellowship Training and Direct Lab Assignment, impacting potential growth over time.
- Respond to in-game events and challenges, such as choosing actions, providing answers, or otherwise reacting to encountered situations, which may include solving riddles or making strategic choices based on the scenario presented.
- Include a main menu. Provide an interactive menu with at least five options that the players see on their turn. At least two of these options should have secondary layers. For example:
    - (1) Check Player Progress:
        - (1) Review Discover Points or (2) Review Trait stats
    - (2) Review Character: Check your character name and experience.
    - (3) Check Position: Display board to view current position.
    - (4) Review your Advisor: Check if the user has an advisor.
        - (1) Display the advisor's abilities or (2) Use the abilities for the challenge.
    - (5) Move Forward: Access this option to spin the virtual spinner.

3.2 Visual Board Representation

Step 1. Set Up the Initial Board Structure

- Use the provided board files, `board.cpp` and `board.h`, to display the default 52-tile trail on a 2-lane path (See Figure 1 and Figure 2).
- Ensure that the starting tile (gray) and ending tile (orange) are visible on both lane paths. You may want to label which path corresponds to Fellowship Training and which path corresponds to Direct Lab Assignment.
- Randomize the tile arrangement. The tiles on the map are the default options but should be randomized each time the game starts. Additionally, ensure that the distribution rules for the tile differ between the two paths, creating unique gameplay experiences for each path. You should design your own rules for the tile generation based on what you think makes sense and create a well-balanced game. You must **randomize each path**, and the distribution rules for each path **must be different**.

- The provided example board currently displays starting tile, regular tile, special tile, and ending tile templates. You need to implement the functionality of each of these tile types on the board (See 3.3).
- The final tile is the Genome Conference. This tile signifies the game's endpoint and should convert each of the Traits to Discover Points to determine and congratulate the winner!

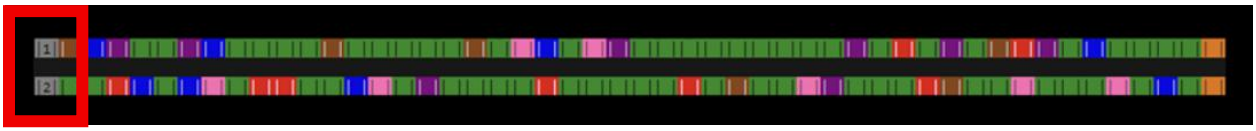


Figure 1. Example of the two-lane board when players choose different path types.

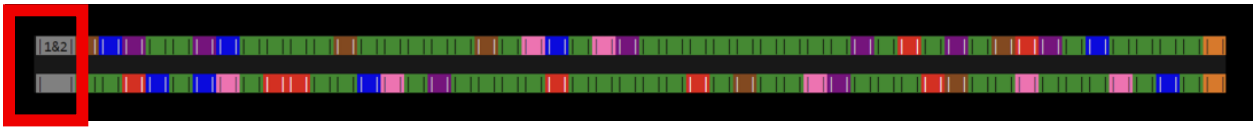


Figure 2. Example of the two-lane board when players choose the same path type.

Step 2. Implement Player Representation on the Board

- Add functionality to visually represent each player's position on the board
  - Use "1" to represent Player 1 and "2" to represent Player 2.
  - The final tile is colored orange, and it represents the Genome Conference.
  - The starting tile is gray and represents that starting point after both players have decided on their path choices.
  - Update each player's position on the board and display the board at the end of every turn. The position should reflect any forward or backward moves resulting from the game spinner or event outcomes.

3.3 Tile Details / Descriptions

Regular Tiles: These tiles represent random events that have probability-based outcomes.

- Regular tiles are default colored green. For each path, ensure there is a minimum of 20 regular tiles on the board. When triggered, randomly select an event from the random.txt file's event list the impacts the player's journey. Implement a 50% chance for an event to be triggered when landing on these regular tiles.

Special Tiles: These tiles are full of surprises. There should be at least 20 special tiles with random positions across each lane of the board. When triggered, events from the special tiles impact the player's Traits. Note that a majority of the special tiles refer to Section 3.4 on DNA sequencing. The special tiles are listed below:

- Blue Tiles: DNA Task 1 – Similarity (Equal-Length)
- Pink Tiles: DNA Task 2 – Similarity (Unequal-Length)
- Red Tiles: DNA Task 3 – Mutation Identification
- Brown Tiles: DNA Task 4 – Transcribe DNA to RNA
- Purple Tiles: Time for a test of wits! Land here, and you'll face a riddle randomly pulled from `riddles.txt` file. Answer correctly, and you'll earn a boost of 500 Points to the Insight Points. Your cleverness pays off!

3.4 Required DNA Sequencing Feature

Your game must implement the following four DNA analysis tasks. Each task will be triggered depending on which tile your character lands on. The player is to provide the input for the corresponding function/DNA task they land on. The function should also influence the players stats.

3.4.1 Background

Before you dive into implementation, let's review a bit of biology to help you understand what you will be working with. DNA, or deoxyribonucleic acid, is the molecule that carries genetic information in almost all living organisms. It acts like a set of instructions that tells cells how to function, grow, and reproduce. DNA is made up of four chemical bases:

- Adenine (A)
- Cytosine (C)
- Guanine (G)
- Thymine (T)

These bases pair together (A with T, C with G) to form the DNA double helix, which is the overall structure of DNA. A DNA strand is made up of a sequence of these base pairs. Every organism has its own unique DNA sequence, through many organisms that share similarities, especially related species. Sometimes, DNA sequences can change, leading to mutations, such as:

- Substitutions (One base is swapped for another)
- Insertions (Extra bases are added)
- Deletions (Bases are removed)

3.4.2 Blue Tiles: DNA Task 1 - Similarity (Equal-Length)

Comparing DNA sequences allows researchers to identify similarities and differences that may be significant in understanding genetic relationships or disease mechanisms. In this step, you'll develop functions to compare DNA strands and assess their similarity. For two DNA strands of equal length, the similarity score is calculated based on the number of matching bases at corresponding positions using the following formula:

$$\text{Similarity} = \frac{\text{total matches}}{\text{total positions}}$$

Example: Let's compare the following two DNA strands:

Position	Strand 1	Strand 2
1	G	G
2	A	T
3	T	T
4	C	C
5	A	A
6	G	A

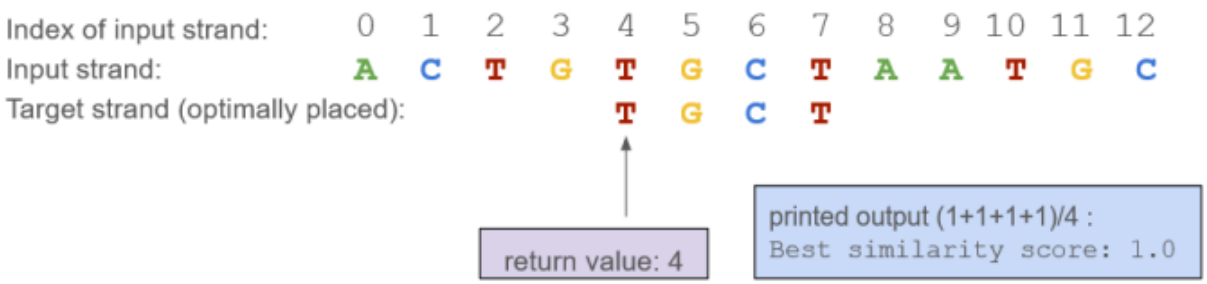
The total number of matches is 4 out of 6 positions, resulting in a similarity score of 4/6 = 0.667.

Function	`double strandSimilarity(string strand1, string strand2)`
Purpose	The function will find the similarity between two DNA strands.
Parameters	`string strand1`: The first DNA strand to compare. `string strand2`: The second DNA strand to compare.
Return Value	The function should return the similarity score between the two strands.

3.4.3 Pink Tiles: DNA Task 2 - Similarity (Unequal-Length)

The strands can be different lengths, therefore, you'll want to compare overlapping sections and calculate similarity scores at each position, and to do that you'll slide the shorter strand along the longer strand. The maximum score across all positions indicates the best alignment between the two strands.

Example Visual:



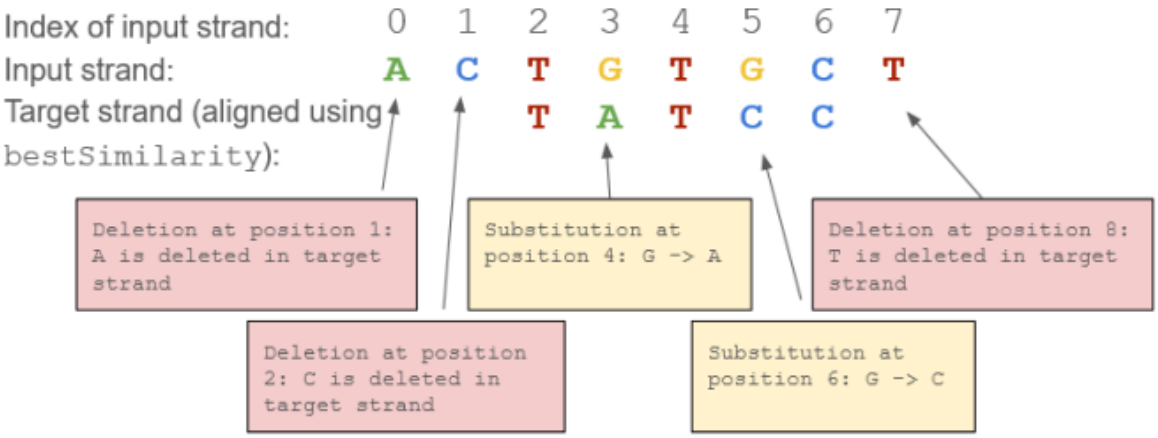
Function	`int bestStrandMatch(string input_strand, string target_strand)`
Purpose	The function will find the best similarity between two DNA strands.
Parameters	`string input_strand`: The input DNA strand to be checked against the target_strand `string target_strand`: The target DNA strand
Return Value	If the parameters are valid, return an int representing the starting index of the substring in the input strand where the best alignment with target strand occurs

3.4.4 Red Tiles: DNA Task 3 - Mutation Identification

Now, you want to be able to make deeper observations between DNA sequences. To do this, you will create a function that compares two DNA sequences and identifies all types of mutations between them. The function should align the sequences based on the best possible match and then process the sequences character by character, printing out mutations as they are detected. Your function should be able to identify the following mutations:

- Substitution: When bases at the same position differ
- Insertion: When an extra base is present in the target strand
- Deletion: When a base from the input strand is missing in the target strand

Example Visual:



Function	`void identifyMutations(string input_strand, string target_strand)`
Purpose	The function compares two DNA sequences to identify all types of mutations between them. It aligns the sequences based on the best possible match and processes them character by character, printing out any mutations as they are detected.
Parameters	`string input_strand`: The input strand to be checked against the target `string target_strand`: The target strand
Return Value	N/A

3.4.5 Brown Tiles: DNA Task 4 - Transcribe DNA to RNA

You will simulate the transcription process by converting a DNA sequence into an RNA sequence. This involves replacing every occurrence of thymine (‘T’) with uracil (‘U’) in the DNA strand.

Function	`void transcribeDNAtoRNA(string strand)`
Purpose	The function will transcribe a DNA sequence to RNA and print the RNA sequence to the console. The function will replace all occurrences of ‘T’ with ‘U’.
Parameters	`string strand`: The DNA sequence to be transcribed.
Return Value	N/A

4 Implementation Requirements

4.1 Class Structures

These classes are strongly encouraged, but not an exhaustive list of classes that may be valuable in your code.

- Board Class: Represents the game board with an array of Tile objects, each with specific properties and types.
- Tile Class: Represent each tile, with attributes for type and references to any potential effects (e.g., challenges, random events, etc.).
- Player Class: Tracks players attributes, including Experience, Accuracy, Efficiency, Insight, and Discover Points. Includes methods for adjusting these attributes based on events.

4.2 Data Members and Methods

- Ensure each class has at least four data members, with appropriate getters, setters, and constructors.
- Use an array of objects from one user-defined class within another.

4.3 File I/O

- Write game stats to a file at the end of the game
- Read files such as `random_events.txt`, `riddles.txt`, and `characters.txt`

4.4 Final Evaluation

- **All players reach final tile**: Calculate each player's total Discover Points. For every 100 points in Accuracy, Efficiency, or Insight, add 1,000 Discover Points to their Discover Points total.
- **Declare the winner**: Display the name of the scientist with the highest Discover Points as the winner, along with each player's final stats

5 Group Work Overview

You are allowed to work with (at most) one other student who is currently in CSCI 1300 this semester (in any section). You may also work by yourself if you choose.

If you choose to work in a group, there are additional requirements. We expect that you will contribute equally to the project. Both group members must submit a zip file for the project,

and the solution files can be the same. Indicate your partner's name in the comments at the top of each code file.

5.1 Group Requirements

In addition to the requirements reviewed above, if you work in a group, you must also implement a sorting algorithm and apply it to a task in your program, and at least one other customization in your code. You should not use a Library function or any external resources to implement the sorting algorithm. Possible customizations are listed in Section 6. One situation where the sorting functionality would be useful is for ranking task, for example ranking all of the players who have completed this game based on their final Discover Points value.

6 Customization

Listed below are some suggestions on how to put your own spin on this project. Note: Any changes you make cannot reduce the complexity of the game or the information in files.

- Game Balancing: Feel free to tweak any numbers in this document to make the game more competitive or interesting.
- Modify the Text Files: You can also modify and add to the text files.
- Game Complexity: You can also shake things up in a more creative way to add complexity to the game. Just ensure that anything you change does not reduce the current complexity of the game or information.
- Change the Theme: The game could take place in an entirely different world or environment.
- Alter the Characters: Instead of scientists, you could have characters from anywhere else with their own attributes.
- Customize the Challenges: Introduce unique challenges that fit your theme. Instead of solving riddles, you can compete in skills-based contests.
- Add Unique Interactions: Think about different ways the players can interact with the game, such as special abilities or items.
- Modify the Game Board: Increase the number of tiles or change tile colors. Change the player representations on the map to symbols rather than numbers. Make sure to stick to the minimum requirement of 52 tiles for each lane and at least two lanes on the board.
- Create More Classes/Structs: You can also add more classes and structs to represent the complexity you want to implement in your game.

7 Timeline

- **Friday, December 5 at 11:49 pm**: Submit all files necessary to compile your code as a zip file.
- **Wednesday, December 10th**: All interview grading will be completed by this date. You must sign up for an interview grading timeslot. If you do not sign up or miss your interview, then no points will be awarded for the project.

8 Scoring

This final project is worth 10% of your grade. The grading interview is required for a grade. Assignments with AI generated code will be given an automatic 0.