Part 2:

Genetic Algorithm Design

Fitness Function:

Fitness = Log (P (team a > team b) + 1) + Log (P (team b > team c) + 1) +… + Log ((team x > team y) +1)

Multiplication of probabilities many times leads to extremely small values. Hence we Apply logarithm to the probability to solve this problem.

As we know that all the probabilities are less than 1 so their log will be negative. Hence to remove negatives, 1 has been added to every probability term.

Crossover Operator:

In the given example, absolute positioning of the teams to be selected is the most important aspect to be considered while choosing the crossover operator. To get an effective crossover operator the key thing is to account for what the properties are in the parents that make them good, and try to extract and combine exactly those properties. An operator that attempts to preserve absolute position in the string is the Order 1 crossover operator.

Hence I have used the Order 1 as the Crossover operator.

Mutation Operator:

The mutation is simple switching of places of two (or more) elements of ranking. Therefore the Mutation operator that I have used is the Swap Mutation.

Basic Terminology:

- Create an array to store the indices of each team in the team's vector in the order that produces the highest probability ranking.
- Create a 2D array to store the probabilities that one team beats another.
- Run the genetic algorithm (Mutation, Crossover and Selection) and store the result in bestRanking.
- Run the cycle of selection, crossover, and mutation until a better total fitness hasn't been found for 100 cycles.
- Output the final ranking.