Concordia University
COEN 432/6321
Winter 2017/18
Assignment 1 (final)

Deadline Part A:	Thursday January 25th, 2018
Deadline Part B:	Thursday February 1st, 2018

Objective: Jana is having her wedding and she is having troubles arranging the table seating plan. We need to solve this problem by using a genetic algorithm to search for the best seating plan possible.

Part A

Design and implement a genetic algorithm to assign guests to seats based on Jana's preferences; who she wants them to sit next to or not next to.
There are five possible preferences for each pair of guests:

Sits next to	= 5
Sits near	= 4
Neutral	= 3
Not next to	= 2
Not near	= 1

"Next to" means on the same table and next to each other on the right or the left side of the seat. "Near" means on the same table only.
"Neutral" means we do not care where they are seated relative to each other.
"Not next to" means not next to each other but they can be on the same table.
"Not near" means not on the same table.

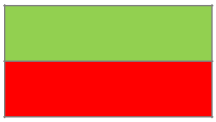
The size of the tables in the wedding is a constant **k** where $\mathbf{5 \leq k \leq 10}$

There are two input files:
settings.txt which has two lines. The first line has **k**: the size of each table. And the second line has **n**: the number of guests in the wedding. Note: the number of tables needed can be determined from that.

guests.csv will have **[n+1] [n+1]** table. The first column **[0][n]** and first row **[n][0]** will have the names of the guests and the rest of the table will represent the preference of each pair from 5 to 1 as described above.

For example, in **guests.csv** file where the number of guests = 5:

	Jasmine	Sarah	Weirdo	Kaylie	Ciara
Jasmine		1	3	3	2
Sarah	3		2	3	1
Weirdo	5	3		3	3
Kaylie	3	3	1		3
Ciara	2	4	3	3	



Weirdo wants to sit next to Jasmine.
Kaylie doesn't want to sit near Weirdo.

The output of the algorithm is a **csv** file that represents a list of the guests' names with the table number and seat number on that table.

For example, "**output.csv**"

Name	Table Number	Seat Number
Jasmine	4	2
Sarah	1	5
…		

Which means Jasmine is sitting on table 4, seat 2 and Sarah is sitting on table 1, seat 5.
The output list can be in any order.

Fitness Evaluation:
Jana knows that a perfect seating arrangement **might be impossible**. However, she decided on the following penalties for the preferences that were not met.

The best fitness is = 0

To calculate the fitness:

penalty = 0
For each Person **P**:
  For each Person **P'**:
    switch (Preference between **P** and **P'**)
      case **1**:
        if **P** is sitting next to **P'** on the right or the left
          penalty += 15
        else if **P** is sitting on the same table as **P'**
          penalty += 10
        break
      case **2**:
        if **P** is sitting next to **P'** on the right or the left
          penalty += 15
        break
      case **3**:
        break
      case **4**:
        if **P** is not sitting on the same table as **P'**
          penalty += 10
        break
      case **5**:
        if **P** is not sitting next to **P'** but they are on the same table
          penalty += 15
        else if **P** is not sitting next to **P'** and they are not on the same table
          penalty += 20
        break

fitness = penalty

Objective: Jana wants to have multiple seating arrangements that will be equally good (same fitness) but different to be able to pick one as she prefers. To do so we need to make sure we have genomic diversity between the fittest individuals (the best seating arrangements).

Part B

In this part, you need to implement a model to enhance on diversity. Choose a model out of the following:

From the book **"Introduction to Evolutionary Computing"** you can refer to:

5.5.3 Fitness Sharing (page 92)
5.5.4 Crowding (page 93)
5.5.5 Automatic Speciation Using Mating Restrictions (page 95)
5.5.4 Running Multiple Populations in Tandem: Island Model EAs (page 95)
5.5.4 Spatial Distribution Within One Population: Cellar EAs (page 97)

You need to compare the diversity between the algorithm in Part A and the algorithm after the diversity enhancement.

To do so, **you need to have everything else exactly fixed**. For example, same size of population, same variation parameters, same number of parents selected, and same selection methods.
The only thing different is the diversity enhancement part.

To measure diversity between seating A and seating B

```
diversity = 0
for each person P from 1 to n
        seatingMatched = false
        positionMatched = 0 //0 for nextToR and 1 for nextToL

        if A.P.nextToR != B.P.nextToR AND A.P.nextToR != B.P.nextToL
                diversity++
        else
                seatingMatched = true

                if A.P.nextToR == B.P.nextToR
                        positionMatched = 0
                else
                        positionMatched = 1

        if !seatingMatched
                if A.P.nextToL != B.P.nextToR AND A.P.nextToL != B.P.nextToL
                        diversity++
        else
                if !positionMatched AND A.P.nextToL != B.P.nextToL
                        diversity++
                else if positionMatched AND A.P.nextToL != B.P.nextToR
                        diversity++

        temp = 0

        for each person P' seated on the same table as A.P
                if P' is seated on the same table as B.P
                        temp++

        for each common empty seat on both tables
                temp++
                //if each table has 2 empty seats temp is incremented twice.
                //if one table have one empty seats and second has no empty seats
                //temp is not incremented

        diversity += (size of table - 1 - temp)
```

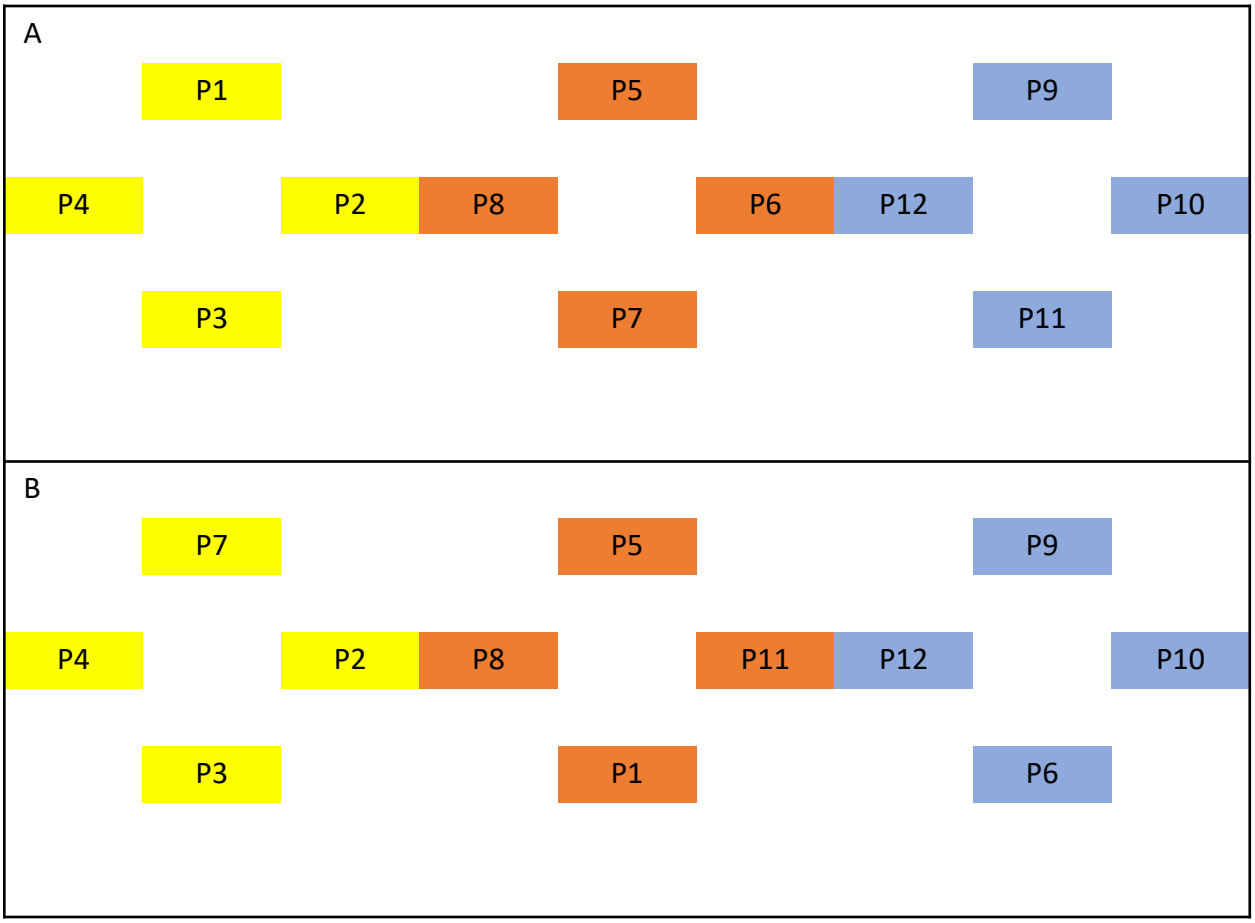
where:
X.P.nextToR: person next to X.P on the right side
X.P.nextToL: person next to X.P on the left side

To measure diversity overall the population at the end of an evolutionary run:

Pick the fittest individuals in the population
Calculate the diversity between them without repetition; Diversity (A, B) is the same as Diversity (B, A)
Sum all the diversity measured

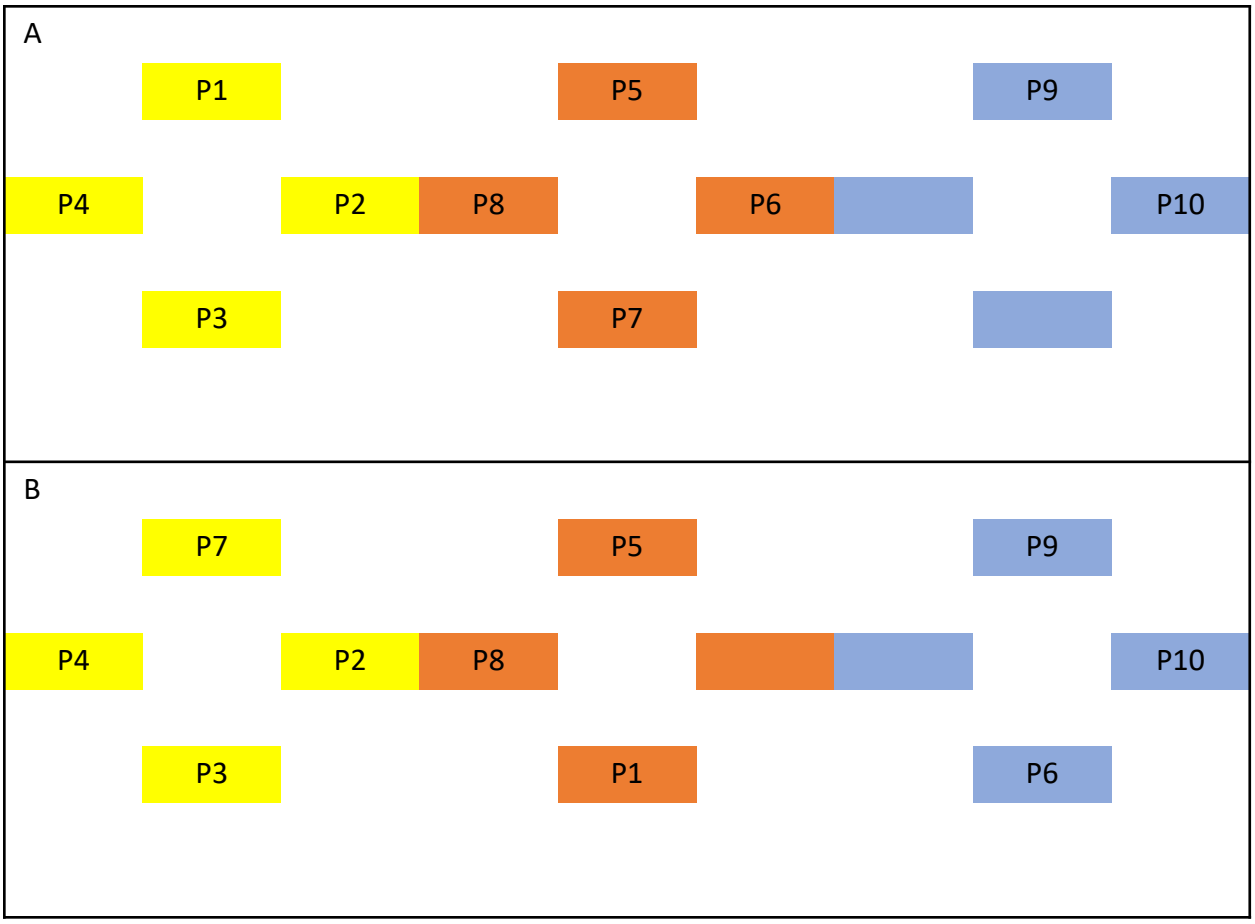
For example, if we have 3 individuals (A, B and C) with fitness = 0
Diversity of the population = Diversity (A, B) + Diversity (A, C) + Diversity (B, C)

Examples for calculating diversity between two seating arrangements:

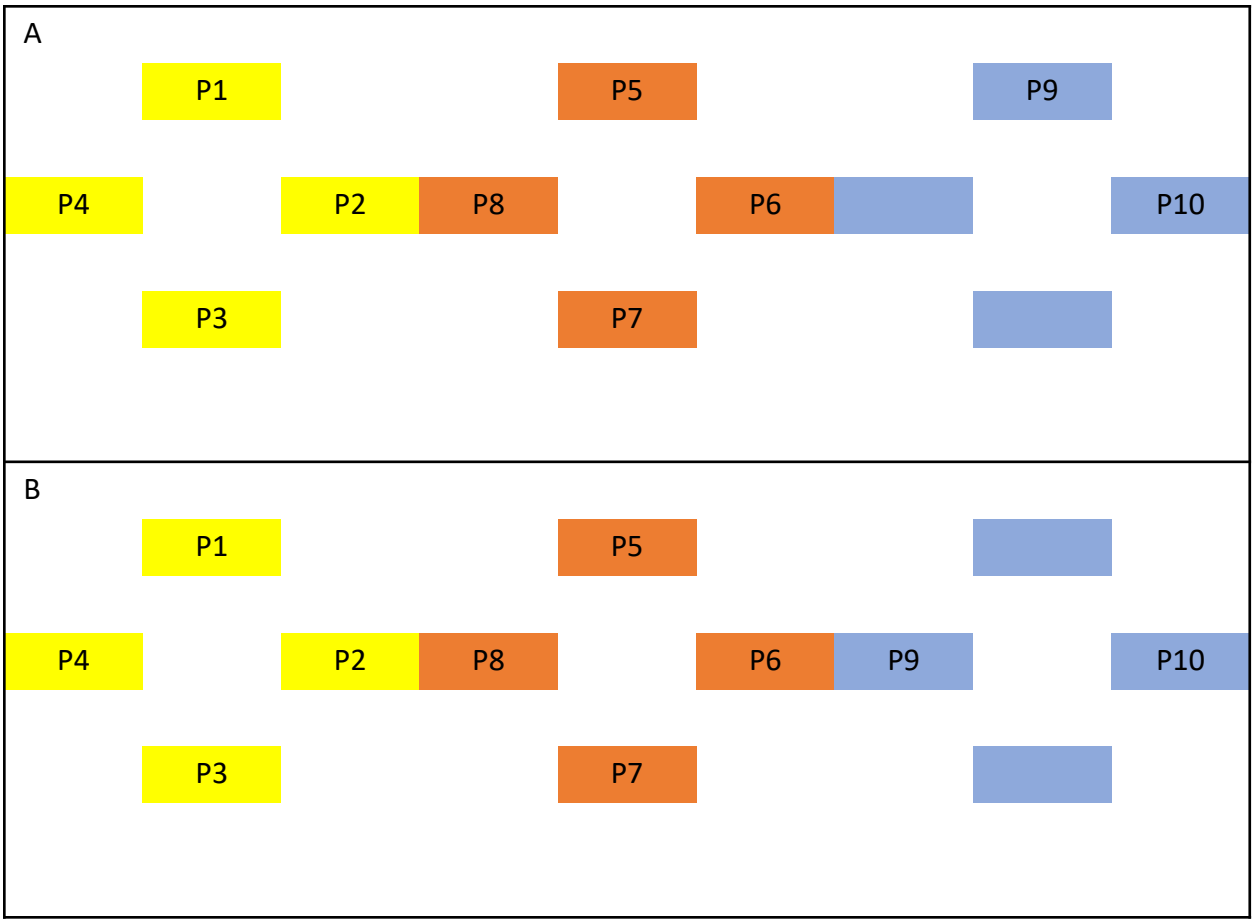


P	A.nextToR	A.nextToL	A . s a m e table	B.nextToR	B.nextToL	B . s a m e table	T o t a l Diversity per person
1	4	2	2,3,4	11	8	5,8,11	5
2	1	3	1,3,4	7	3	7,3,4	2
3	2	4	1,2,4	2	4	7,2,4	1
4	3	1	1,2,3	3	7	3,2,7	2
5	8	6	6,7,8	8	11	1,8,11	3
6	5	7	5,7,8	10	12	9,10,12	5
7	6	8	5,6,8	4	2	2,3,4	5
8	7	5	5,6,7	1	5	1,5,11	3
9	12	10	10,11,12	12	10	10,6,12	1

10	9	11	9,11,12	9	6	6,9,12	2
11	10	12	9,10,12	5	1	1,5,8	5
12	11	9	9,10,11	6	9	6,9,10	2
							36



P	A.nextToR	A.nextToL	A.same table	B.nextToR	B.nextToL	B.same table	Total Diversity per person
1	4	2	2,3,4		8	5,8	5
2	1	3	1,3,4	7	3	7,3,4	2
3	2	4	1,2,4	2	4	7,2,4	1
4	3	1	1,2,3	3	7	3,2,7	2
5	8	6	6,7,8	8		1,8	3
6	5	7	5,7,8	10		9,10	5
7	6	8	5,6,8	4	2	2,3,4	5
8	7	5	5,6,7	1	5	1,5	3
9		10	10		10	10,6	1
10	9		9	9	6	6,9	2
							29



P	A.nextToR	A.nextToL	A.same table	B.nextToR	B.nextToL	B.same table	Total Diversity per person
1	4	2	2,3,4	4	2	2,3,4	0
2	1	3	1,3,4	1	3	1,3,4	0
3	2	4	1,2,4	2	4	1,2,4	0
4	3	1	1,2,3	3	1	1,2,3	0
5	8	6	6,7,8	8	6	6,7,8	0
6	5	7	5,7,8	5	7	5,7,8	0
7	6	8	5,6,8	6	8	5,6,8	0
8	7	5	5,6,7	7	5	5,6,7	0
9		10	10			10	1
10	9		9			9	1
							2

A

	4			14			8		
2		3	13		11	6		7	
1		5				9		10	
	0			12					

B

	4			14			3		
5			12		11	10		2	
		0	13		8	9		6	
	1						7		

Diversity between seating A and seating B = 57

Documentation:

The documentation will have the following sections:

A. Part A
    1. The Genetic Algorithm *(3.2 Components of Evolutionary Algorithm)*
        - Representation
        - Fitness Evaluation
        - Population
        - Parent Selection Mechanism
        - Mutation and Crossover Operators
        - Survivor Selection Mechanism
        - Initialization
        - Termination Condition
    2. Testing
       Define all the testing that was done with result. You need to generate your own data to test the algorithm on different inputs.
B. Part B
    1. Diversity Enhancement
       Describe the model you chose and how it works.
    2. Comparison between Part A diversity and Part B diversity
    3. Testing (same as above)
C. Installation and Running Instructions

Submission:

There will be 2 submission links on **moodle** for each part of the assignment. One for the **source code** and another for the **pdf document**.

The source code and report for Part B submission needs to include the full project!

For the source code: you need to clean the project from any builds before submitting. The file submission must be **.zip** file named **StudentID_Ass1** containing your source code project.

For the document: you need to submit a **pdf** file named **StudentID_Ass1**.

TJ

Grading Scheme:

The objective of grading is to fairly evaluate your assignment in regard to the assignment requirements. Your program must, first and foremost, be functional, i.e. **if it doesn't work, then you've failed the basic requirement and the assignment will not receive a mark**. Part of an evolutionary algorithm course, the fitness of at least the fittest individual must improve in comparison to the initialized random individuals.
We will ask you to demonstrate your code on a test guest list *.csv file and study your output – must be formatted as instructed.
Assignments are to be submitted individually. You may and are encouraged to discuss with fellow classmates, but plagiarism is gravely in tolerated.

Grade Breakdown:

- Accounts for 15 marks out of: 75 for undergrads, 100 for grads of final grade
- Does it satisfy the bare minimum requirements? If yes, continue; or else, assign *0* and stop.

[20%] Documentation
- Evaluation: structure of report, well written, clearly explained process, discussion of results either good or failed. Basic spell-check, grammar and syntax. Your report should read like a chronological story. References.

[30%] Discussion
- Evaluation: The individual representation (called the genotype) must be explained and demonstrated: why did you pick such and such representation? Does it handle phenotypical invariances (yes, because; no, but why not)? Does it simplify fitness evaluation? How is cross-over and/or mutation applied? Which cross-over and/or mutation operations were selected and why? Please state and discuss your thought process and list your assumptions.

[20%] Diversity Measures and Implementation
- Evaluation: how do you maintain diversity? How is it measured? Does the method work as compared to the baseline (Part A)?

[20%] Live Demonstration and Evaluation
- Evaluation: your algorithm will go head to head with ours. Given a test set, you will demonstrate your code converging to a solution. Your fittest individual(s) will be compared to ours. Depending how close they are to ours, a mark will be assigned.

[10%] Innovative and Elegant Solutions
- Evaluation: at our discretion.

VD