TCP1231 Computer Programming I

Trimester I, Session 2009/2010
Faculty of Information Technology
Multimedia University

Project II (15%)
Alien and Predator Friendly Match

Section 1. Introduction

Alien and Predator have been enemy for decades. Both parties hold many wars before, but none of them win without sacrificing their soldiers. One day, the director of Predator had contacted the director of Alien to discuss ways to stop the war. But they insisted to distinguish who is the most powerful one. After a few discussions, they decided to hold a friendly match. This time, no party is allowed to hurt the other party physically. The friendly match is a numbering game. The one who achieves the highest score will be the winner.

You and your partner are elected as the middle persons to create the game. The game should be playable by two parties. Each party only allows one representative to participate in the game.

Objective

The objective of this project is to develop a game which is playable by two persons. Each will take turn to select their choices.

Use vector class or create a huge array to store the auto-generate randomized numbers. The two players will take turn to choose the numbers one at each turn. Once the numbers in any row or column are fully chosen, the game is considered ending and the winner will be the one who obtains the highest score. Please refer to **Section 5** for more details of the rules and tasks to be accomplished by you and your partner in the program.

Section 2. Deliverables

Students must **EMAIL** their **report** (softcopy of course) and the **program** (.cpp file) to your **RESPECTIVE** tutor. Your tutor will then reply to confirm the acceptance of your email. If you do not receive any email confirmation, please phone or email to him/her again.

Since this is a two persons' work, only one person needs to email the documents to us. It is his/her partner's responsibility to make sure that their work is sent by his/her group-mate to the tutor. No claim will be accepted that he/she does not know his/her partner did not submit the documents.

Below are the tutors' email addresses and contact numbers **ACCORDING** to your lab section:

TC201 - Mr. Ban Kar Weng	(workjudge@gmail.com, 03-83125267)
TC202 - Mr. Victor Soh	(cmsoh@mmu.edu.my, 03-83125267)

TC203 - Mr. Ng Kok Why	(kwng@mmu.edu.my, 03-83125244)
TC204 - Mr. Edwin Law	(edwin@mmu.edu.my, 03-83125277)
TC205 - Mr. Ooi Wen Fong	(wfooi@mmu.edu.my, 03-83125205)
TC206 - Ms. Yang Chi Shian	(csyang@mmu.edu.my, 03-83125391)
TC207 - Mr. Lee Kian Chin	(kclee@mmu.edu.my, 03-83125343)
TC208 - Mr. Lee Kian Chin	(kclee@mmu.edu.my, 03-83125343)
TC209 - Mr. Ng Kok Why	(kwng@mmu.edu.my, 03-83125244)

In the REPORT, students are required

- **To write down the extra features** (must be as brief as possible, e.g. one sentence for each extra feature) that they have included in the project.
- To show some screenshots of the program (with clear description) and design a simple user manual to guide the examiner to manipulate your program.
- To attach together this document – **Project Mark Sheet (**pg. 9 in this file)
- Do not put the whole source code in the report. If you wanted to show some special functions, you may just include those functions alone in the report followed by some explanation.

Please refer to **Section 9** for more details on the submission instruction.

Section 3. Deadline

This project must be submitted by **8 September 2009 (Tuesday), 9.00am** to your respective tutor. Please note that submission of the project is **COMPULSORY** for this course. Late submissions will be penalized 20 marks for each 24 hours (1 day) late.

From now on, you have plenty of time to construct your program. Start your work as early as possible. Do not wait until last minute. If you start early, you can consult and get help from your lecturer or tutor. Last minute work will only make you rush, and your lecturer and tutor own their full right to reject your last minute request for help on problems related to earlier-taught lessons.

Section 4. Group work

This project is to be done in a **GROUP** (not more than **TWO** (2) persons from the **SAME LAB** section. **Doing it ALONE** is also allowed.). **STRICTLY NO COPYING** from other students or from other sources (from the Internet, books, etc.). **Plagiarism and cheating is an offence** in MMU, and you can be failed for this subject and be reported to the Faculty for action.

Section 5. Rule of the Game (Maximize the Total)

In the initial stage of the game, the program shall request one of the players to **enter the size** of a grid that they want to play. For example, the input size is 6. So, a 6x6 grid will be generated. This should be followed by a list of **auto-generate randomized numbers** to fill up the blanks (see the example below). The numbers do not need to be **unique** and **can be repeated**. Below the grid is the **score owned by player1 and player2**. Of course the started score is zero for both players.

Example:

```
---------------------------------------------
| 18  |  9  | 15 | -20 | 22 | -7 |
---------------------------------------------
| 22  | 24 | 15 |  -5  | -13 | 18 |
----------------------------------------------
|-13  |  1  | 21 | 10  | -4  |  6  |
----------------------------------------------
|  13  | -7  | -16 | -19 | -4  | -2  |
----------------------------------------------
| -19 | 22 | 24  | 13 |  9   | 3  |
----------------------------------------------
| -17 |  9  | 15  | -16 | -8  | -7  |
----------------------------------------------
     Player 1: 0          Player 2: 0
```

<u>In 1st step:</u>

Make the PC to randomly select a starting row or column, then 1st player is to select his prefer number from the grid. Example, the PC selected 2nd row in the grid. Let say the 1st player selected -5. The value will then be added to 1st player's score and the selected number will turn into "x".

Example:

```
---------------------------------------------
| 18  |  9  | 15 | -20 | 22 | -7 |
---------------------------------------------
| 22  | 24 | 15 |   x  | -13 | 18 |
----------------------------------------------
|-13  |  1  | 21 | 10  | -4  |  6  |
----------------------------------------------
|  13  | -7  | -16 | -19 | -4  | -2  |
----------------------------------------------
| -19 | 22 | 24  | 13 |  9   | 3  |
----------------------------------------------
| -17 |  9  | 15  | -16 | -8  | -7  |
----------------------------------------------
    Player 1: -5          Player 2: 0
```

<u>In 2nd step:</u>

Now, it is the 2nd player's turn. He can only select any number on the same **COLUMN** with the previously selected number. In the case above, he can only consider numbers in 4th column (which are -20, 10, -19, 13, -16).

Suggestion for the 2nd player's selection: Since the condition to win the game is to get the highest score, the 2nd player shall choose the largest number from the grid. In that case, number 13 will be the choice (see the example below). Remember to update the score for 2nd player.

Example:

```
---------------------------------------------
| 18  |   9  |  15 |  -20 |  22 |  -7 |
---------------------------------------------
| 22  |  24 |  15 |    x  | -13 | 18 |
----------------------------------------------
|-13  |   1  |  21 |  10  |  -4  |  6  |
----------------------------------------------
|  13  | -7  | -16 | -19 |  -4  | -2  |
----------------------------------------------
| -19 |  22 |  24  |   x  |   9   |  3  |
----------------------------------------------
| -17 |   9  |  15  | -16 |  -8  | -7  |
----------------------------------------------
      Player 1: -5          Player 2: 13
```

<u>In 3rd step:</u>
It is back to 1st player's turn now. The player can only choose the numbers in the same **ROW** with the previously selected number by 2nd player. In this case, the 1st player can only consider the numbers -19, 22, 24, 9 and 3. Of course the largest number here will be 24. Add the number to 1st player's score. Anyhow, this is just a suggestion. It is up to the player to decide which number he likes.
Example:

```
---------------------------------------------
| 18  |   9  |  15 |  -20 |  22 |  -7 |
---------------------------------------------
| 22  |  24 |  15 |    x  | -13 | 18 |
----------------------------------------------
|-13  |   1  |  21 |  10  |  -4  |  6  |
----------------------------------------------
|  13  | -7  | -16 | -19 |  -4  | -2  |
----------------------------------------------
| -19 |  22 |   x   |   x  |   9   |  3  |
----------------------------------------------
| -17 |   9  |  15  | -16 |  -8  | -7  |
----------------------------------------------
      Player 1: 19          Player 2: 13
```

<u>In 4th step:</u>
Repeat 2nd step above for 2nd player's turn. It is actually alternating between **COLUMN** and **ROW**.

<u>Ending condition:</u>
The game will be ending if there is any row or column **fully chosen** (or fully turn into "x" as shown in red box) below.

Example:

```
-------------------------------------------
|  x  |  x  |  x  | -20 |  x  | -7 |
-------------------------------------------
|  x  |  x  |  x  |  x  | -13 |  x  |
--------------------------------------------
| -13 |  1  |  x  |  x  | -4  |  6  |
--------------------------------------------
| 13  | -7  |  x  | -19 | -4  | -2  |
--------------------------------------------
| -19 |  x  |  x  |  x  |  x  |  3  |
--------------------------------------------
| -17 |  x  |  x  |  x  | -8  |  x  |
--------------------------------------------
    Player 1: 127        Player 2: 95
```

At the end, the program will report by either "Congratulation, the 1st player has won!" or "Congratulation, the 2nd player has won!!" for the players. You may change the name of the players to Alien and Predator.

Some more tip for selecting the number: Sometime (in a smarter way), the player may not necessary to always choose the largest number in a row or column. He may choose other number (not the largest) which would cause his opponent to choose other smaller or negative numbers. With that, his opponent's score would be reduced. See the example below for 2nd player's selection (I have slightly modified the numbers below to serve the purpose of the smarter way above).
Example: If you choose 13 in 4th column, your opponent will choose 24 (which is more advantage than you). But if you choose 10, your opponent will have to choose -3 (which is disadvantage to your opponent).

```
-------------------------------------------
| 18  |  9  | 15  | -20 | 22  | -7  |
-------------------------------------------
| 22  | 24  | 15  |  x  | -13 | 18  |
--------------------------------------------
| -13 | -3  | -21 | 10  | -4  | -6  |
--------------------------------------------
| 13  | -7  | -16 | -19 | -4  | -2  |
--------------------------------------------
| -19 | 22  | 24  | 13  |  9  |  3  |
--------------------------------------------
| -17 |  9  | 15  | -16 | -8  | -7  |
--------------------------------------------
    Player 1: -5        Player 2: 0
```

Section 6. Program Requirements

1. You are forbidden to use '**goto**' in your program. **ZERO** marks will be given if a single 'goto' is used.
   Global variables are not encouraged. Students will definitely get significant less mark if they use them.

2. Use vector class or create a huge array to store the auto-generate randomized numbers.

3. When the player chooses for the number (let say 24 as shown in red box below), the program shall request the player for the input.

   Suggestion:
   Since in each turn, either the column or the row is being fixed based on previous move of the opponent player, the next player only needs to select EITHER the column or the row number. Thus, your program shall do the following request:

   Current Row is E, select Column →

   or

   Current Column is 3, select Row →

   Or if you have better/direct method to ease the player for the input, you may apply your own method.

```
        1     2     3     4     5     6
      -------------------------------------------
  a  | 18  |  9  | 15  | -20 | 22  | -7  |
      -------------------------------------------
  b  | 22  | 24  | 15  |  -5 | -13 | 18  |
      -------------------------------------------
  c  | -13 |  1  | 21  | 10  | -4  |  6  |
      -------------------------------------------
  d  | 13  | -7  | -16 | -19 | -4  | -2  |
      -------------------------------------------
  e  | -19 | 22  | 24  | 13  |  9  |  3  |
      -------------------------------------------
  f  | -17 |  9  | 15  | -16 | -8  | -7  |
      -------------------------------------------
          Player 1: 0        Player 2: 0
```

   If the player does any **wrong input**, notify the mistake and request him/her to input the value again.

4. Once the number has been selected, turn the number into “x” and add the value to the player’s score. You may erase the number by not displaying it in the scene.

5. After each selection, check the row/column whether it is fully selected. If **YES**, end the game and determine who the winner is. If **NOT**, continue the game.

   When the game is ended, ask the player whether want to continue or to quit the game.

6. There are many additional features you can add to your program. However, these extra features are **NOT** a requirement of this assignment, but you will be rewarded bonus marks if you are able to come up with some. Moreover, most of these special features can make your game more exciting and appealing! See **Section 8** for more on bonus marks, and suggestions of some additional special features you can implement.

Section 7. Evaluation Criteria

This assignment will be evaluated based on the following key items:

- General efficiency of program
- Modularity of program (good and proper usage of functions and arrays)
- Accuracy and error-free
- Organization and structure
- Usability
- Clarity and style (of programming)

Please refer to the Project Mark Sheet on the detailed marks allocation.

Section 8. Bonus

You will be given bonus marks (maximum of extra 10 marks of the total marks of 100) if you are able to incorporate some additional special features to your program. Here are some suggestions:

- Allow the player to **undo** his selected numbers by reversing back to a limit of two numbers from his current selection. Example: The player has selected 24, 10, 5, 7 and 14. The player is allowed to undo his selection backward to 24, 10 and 5. Figure out where and how you can perform this in your program. Please ensure your program remains user-friendly.

  The players can only undo once throughout the game.

- When the players select the number, **LIMIT** the players from only able to select numbers from a row or a column. You may do this by disable the other unrelated rows/columns or to highlight the available numbers for selection.

- Any other significant work that will make the program more interesting, attractive and user-friendly. Feel free to include them into your program. Remember to mention them in your report as well!

Section 9. Submission Instructions

1. Submission Instructions:

a. *Email Title:* **Project02 Submission**

b. Name the *program* and the *report* with **Sender's name** and **ID** (the student who responsible to email the solution to the tutor). Follow the format below.

```
TC20x_10xxxxxxxx_SenderName.cpp and TC20x_10xxxxxxxx_SenderName.doc
```

which

- TC20x – Type your lab section here. Example: TC201
- 10xxxxxxxx – Type your student ID number here.
- SenderName – Type the sender name here with **underscore** to separate the words. Example: Ho_Chin_Kuan. Use your **ICEMS name**, so that we can easily detect your name when we key in the mark into the name list.

c. *Place **ONLY** your **.cpp** file into the folder. **DO NOT** place your **.exe** file and **.txt** file into the folder.*

d. *Zip your .cpp file and email it to your respective tutor (Please refer to **Section 2** above for the email address).*

e. *Late submission will be detected through your lecturers' PC system clock.*

2. Additional Information

   a. Marks will be deducted if the submission instructions are not followed.

   b. You are expected to make sure that your code is easily readable. Pay attention to indentation.

   c. In your .cpp file, please insert appropriate comments to help the lecturers/tutors to understand your code.

   d. Please insert the following info on top of your .cpp file. Please make sure your submission is a **finalized version. If you email (your program) more than one version, only the first email will be entertained**.

```
/*************************************************
Program:  TC20x_10xxxxxxxx_SenderName.cpp
Course:   TCP 1231
Name:     (YOUR_FULL_NAME) and (YOUR_GROUP_MATE_NAME)
ID:       (You and your group-mate's MMU ID)
Lecture:  TC10?
Lab:      TC20?
Email:    (You and your group-mate's email address)
Phone:    (You and your group-mate's phone no.)
*************************************************/
```

TCS1231 Computer Programming I

Project Mark Sheet

To be filled by Examiner. This is for your reference only.

Description	*Max.*	Actual Marks
1. Report (10%)		
a. Screenshots	5	
b. User Guide / Instructions for Program	5	
2. Coding (50%)		
a. Code Efficiency and Strategy	20	
b. Code Modularity (usage of functions)	15	
c. Error Checking Features	10	
d. Style – Self Documentation	3	
e. Style – Indentation	2	
3. Program Execution (40%)		
(If the program cannot run, even if it can compile, 0 mark will be given for this section)		
a. General Appearance of Program (display)	10	
b. Accuracy (program works and runs correctly)	10	
c. Usability (interaction with user, user-friendliness)	10	
d. Performs All Required Features	5	
e. Error-free During Runtime	5	
4. Bonus (max. 10%)		
Additional special features OR any other significant contributions	max. 10	
TOTAL	**100**	