C++ Programming Mini-Project – 2

Lucky888

Practical Work

By using Microsoft Visual Studio 2010, write a C++ object-oriented program to implement the game Lucky888. The player can either start a new game or continue a previous game by loading the player's profile. A new game will assign 100 credits to the player initially; otherwise, the number of credits will be loaded from the profile. In each turn, a hand of poker cards will be distributed for every 20 credits paid, where no identical cards (same rank and same suit) will appear in the same hand. The player is allowed to swap cards. After the player has finished selecting all the cards he/she wants to swap, a "SWAP" button should be clicked and the new cards (must be different from the cards being replaced) are distributed to replace the selected card(s). After swapping has been done, the player should click an "EXTRA" button to get extra card(s) for the final hand. The player will win credits if the final hand hits either one of the following combination:

Combination	Meaning	Credits won
Royal Flush	A, K, Q, J, 10; all of the same suit.	10000
Straight Flush	Five cards in sequence; all of the same suit.	4000
Four of a Kind	Four cards of one rank; plus an unmatched card.	800
Full House	Three cards of one rank, and two cards of another rank.	200
Flush	Five cards of the same suit; not in rank sequence.	150
Straight	Five cards in sequence, but in more than one suit.	100
Three of a kind	Three cards of the same rank, and two unmatched cards.	60
Two Pairs	Two cards of the same rank, two cards of another rank; plus one unmatched card.	40
One Pair	Two cards of the same rank; plus three other unmatched cards.	20

The game ends when the number of credits left is not sufficient to pay for a new set of cards, or the player click SAVE to quit the game. In the former case, the player's profile will be removed. In the latter case, the credits left will be stored in the player's profile.

Reference:

- http://en.wikipedia.org/wiki/Straight_flush#Straight_flush

The number of cards distributed, the number of cards allowed to be swapped and the number of extra cards should follow one of the rules below. The rule number should be determined by the remainder obtained from dividing your team number by 5. For example, if your team number is 6, 6%5 = 1; you should follow rule 1) of this project.

Rule:

0) In each turn, a hand of 5 poker cards will be distributed. The player is allowed to swap any number of cards for one time. Whenever a card is selected by clicking it, the card is indicated by "HOLD". The player can unselect the card by clicking the card once more. After swapping has been done, the player should click an "EXTRA" button to get 1 more card for the final hand.
1) In each turn, a hand of 5 poker cards will be distributed. The player is allowed to swap at most THREE cards for one time. Whenever a card is selected by clicking it, the card turns face-down. The player can unselect the card by clicking the card once more. After swapping has been done, the player should click an "EXTRA" button to get 1 more card for the final hand.
2) In each turn, a hand of 5 poker cards will be distributed. The player is allowed either to swap ALL cards or none for one time. If the player decides not to swap any card, he/she can directly

click the "EXTRA" button. On clicking the "EXTRA" button, the player will get 1 more card, which is allowed to be swapped for one time to form the final hand.

3) In each turn, a hand of 4 poker cards will be distributed. The player is allowed to swap at most TWO cards for one time. Whenever a card is selected by clicking it, the card is indicated by "HOLD". The player can unselect the card by clicking the card once more. After swapping has been done, the player should click an "EXTRA" button to get 2 more cards for the final hand.
4) In each turn, a hand of 3 poker cards will be distributed. The player is NOT allowed to swap any card at the moment. Then, the player should click an "EXTRA" button to get 3 more cards. The player is allowed to swap any number of cards for one time among the 3 extra cards. Whenever a card is selected by clicking it, the card turns face-down. The player can unselect the card by clicking the card once more. After clicking the "SWAP" button, the final hand is obtained.

Example screen shot. (Note that the interface may be different.)

1. Your program needs to be designed with a graphical user interface (GUI).

2. While you are responsible for the final design of the program, it is required that the code governing the rules of the game should be implemented with a separate static library developed by native C++ and linked into the managed C++ application. The managed C++ code you write should mainly be responsible for the GUI.

3. Before a user starts to play, he/she needs to login the system first. A file is then created to store the result of this user when playing the game, e.g. the scores he/she gains for the game. If this user has played this game before, his/her historical record should be retrieved and shown on the screen. After he/she plays the game, his/her record should then be modified and stored.

4. Should you want to get a credit, you should design the game with some advanced features as follows.
   - The player is allowed to pay more than 20 credits for a new set of cards. The credits won will then be proportional to the credits paid.
   - Introduce a joker card as wildcard that can represent any card.
   - If the player wins some credits in one turn, he/she is allowed to use the credits won to play a Double-or-Nothing game, say guessing big or small for a card. If the player wins this game, the credits would be doubled; otherwise, the credits won would be lost.

5. Should you want to get a distinction, you are free to add two more features to the game that will make it to be more interesting. You may seek the opinion of your subject lecturer/tutor before you do so. In principle, it must be some new features that require additional efforts but not a repetition of work.

Report

Your report should include:

1. **Abstract**: An abstract of less than 200 words that summarizes the objectives and achievement of your project.

2. **Introduction**: A detailed description of the objectives and requirements of the project, and a brief description of the methodology.

3. **Methodology**: It contains
   - How your group divides the work among the group members (very important, to be used as the basis for assessment)
   - The schedule of implementing the project
   - The structure of the program developed, including
     - The specifications of the classes defined, and the public/private member functions/variables inside - explain as far as possible why your group makes such choices
     - The flow of execution. (It is good to include a flow chart to illustrate it.)
   - What problems your group encounters, and how your group tackles the problems
   - Testing of your program, which shows
     - How you validate your program, i.e. confirm that the solution is correct.

4. **Results**
   - Include the results of executing your program captured from the screen.

5. **Conclusion and future development**
   - Summarize the experience gained in the mini-project
   - Indicate how your program can be extended and improved if more time is allowed.

In addition, a user manual of your program should be attached to the report as an Appendix to illustrate the usage of your program. It may include, for example, the screen shots of the GUI you developed when running the program.

The report should be in PDF format. It is NOT required to include the complete source code in the report. Rather, you should copy the folder(s) containing all your project files into a CD, which also stores the report. (See the General Description below.)

General Description

1. Each group should comprise 2 students. Students need to obtain prior approval from the subject lecturer if they want to form a group with fewer or more group members.

2. After finishing the project, each group should hand in one CD that contains the following:
   - The soft copy of the report (including the Appendix of the user manual)
   - All folder(s) and files of your project
   - A readme file (`readme.txt`) that tells us how to build the project and, if any, other important (IT) requirements for running the program.

3. You have to make a presentation to demonstrate and explain the details of your work. In the presentation, each group member should declare his/her responsibility in the project. Each member will be individually assessed based on the declared responsibility and the result obtained.

4. The documentation for your mini-project is a very important part. The ability of writing good comments will also be an important factor to the final assessment of your mini-project.

5. It is compulsory to use a word processing tool to write your report. The font size must not be bigger than 12 or smaller than 10. Print your report in 1.5 lines spacing on both sides of a page. Including all diagrams and tables, if any, the length of the report should not be shorter than 15 pages.