|  |
| --- |
| Lab 08, 09 Stack and Queue |
|  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Question 1: Postfix calculator**  Some calculators require you to enter postfix expressions. Recall that an operator in a postfix expression applies to the two operands that immediately precede it. Thus, the calculator must be able to retrieve the operands entered most recently. The ADT stack provides this capability. In fact, each time you enter an operand, the calculator pushes it onto a stack. When you enter an operator, the calculator applies it to the top two operands on the stack, pops the operands from the stack, and pushes the result of the operation onto the stack. The final result will be on the top of the stack.  Use the pseudocode algorithm given below to evaluate postfix expressions. Use only the operators +, –, \*, and /. Assume that the postfix expressions are syntactically correct.  **for** (each character ch in the string) {  **if** (ch is an operand)  Push value that operand ch represents onto stack  **else** { // ch is an operator named op  // evaluate and push the result  operand2 = top of stack  Pop the stack  operand1 = top of stack  Pop the stack  result = operand1 op operand2  Push result onto stack  }  }  **Task:**  Implement the calculator that evaluates a postfix expression by filling out the empty method. Make sure your method can process any number of digits from an integer.  **Sample Run 1:**  Enter a postfix expression to be evaluated: 2 10 100 / \* 1000 +  2 10 100 / \* 1000 + = 1000.2  **Sample Run 2:**  Enter a postfix expression to be evaluated: 2 3 4 + \*  2 3 4 + \* = 14  **Sample Run 3:**  Enter a postfix expression to be evaluated: 2 9 2 / \* 8 + 7 - 4 /  2 9 2 / \* 8 + 7 - 4 / = 2.5   |  | | --- | | **Question** **2: Tower of Hanoi**  Implement a Tower of Hanoi game using ADT Stacks. Allow the user to set up the game with one to five disks. Below is a sample output.  A : 3 2 1  B :  C :  Move from ==> |   **Question 3: Animal Card Game**  Animal Card Game is a simple 2-player card game that consists of 6 cards (2 Mouse cards, 2 Snake cards, and 2 Elephant cards). Each animal is superior to another animal based on the following rule:   1. **Elephant** is superior to **Snake** 2. **Snake** is superior to **Mouse** 3. **Mouse** is superior to **Elephant**   Initially, each player holds three cards. These three cards may not be unique (i.e. repeated animals is possible). At each round of the game, both players display their first card. The player whose first card is more superior captures his opponent's first card, which is done by keeping his opponent's first card at the bottom of his pile of cards (lose card), followed by his first card (win card).  If both of their first cards are the same (i.e. they draw), their first cards will be put in the standby location. Then, both players will keep displaying their next first card until one player captures the other. When that happens, all the cards in the standby location will belong to the winning player. Then, his opponent's first card will again be put at the bottom of his pile of cards, followed by his first card. The game ends when one of the player losses all his cards, or when both players has no cards left (i.e. it's a tie).  Some examples of the game are provided as follows:  **Example 1:**   |  | | --- | | (M = Mouse, S = Snake, E = Elephant)  **Initial Setup:**  Player 1: S, M, E  Player 2: E, M, S  Standby:  **1st Round:**  Player 2 Elephant captures Player 1 Snake  After capture,  Player 1: M, E  Player 2: M, S, S, E  Standby:  **2nd Round:**  Player 1 and Player 2 draws.  Both Mouse cards are put in standby location.  After draw:  Player 1: E  Player 2: S, S, E  Standby: M, M  **3rd Round:**  Player 1 Elephant captures Player 2 Snake.  All cards in standby location belong to Player 1.  After capture,  Player 1: M, M, S, E  Player 2: S, E  Standby:  **4th Round:**  Player 2 Snake captures Player 1 Mouse  After capture,  Player 1: M, S, E  Player 2: E, M, S  Standby:  **5th Round:**  Player 1 Mouse captures Player 2 Elephant  After capture,  Player 1: S, E, E, M  Player 2: M, S  Standby:  **6th Round:**  Player 1 Snake captures Player 2 Mouse  After capture,  Player 1: E, E, M, M, S  Player 2: S  Standby:  **7th Round:**  Player 1 Elephant captures Player 2 Snake  After capture,  Player 1: E, M, M, S, S, E  Player 2:  Standby:  **Player 1 wins.** |   **Example 2:**   |  | | --- | | (M = Mouse, S = Snake, E = Elephant)  **Initial Setup:**  Player 1: E, M, S  Player 2: E, M, S  Standby:  **1st Round:**  Player 1 and Player 2 draws.  Both of the Elephant cards are put in standby location.  After draw,  Player 1: M, S  Player 2: M, S  Standby: E, E  **2nd Round:**  Player 1 and Player 2 draws.  Both of the Mouse cards are put in standby location.  After draw,  Player 1: S  Player 2: S  Standby: E, E, M, M  **3rd Round:**  Player 1 and Player 2 draws.  Both of the Snake cards are put in standby location.  After draw,  Player 1:  Player 2:  Standby: E, E, M, M, S, S  **Tie.** |   **Tasks:**  Write a program that accepts the cards for both players as shown in the sample runs below. Based on the input, the program should show “**Player 1 wins.**”, “**Player 2 wins.**”, or “**Tie.**”. Remember to remove [App\_q1.cpp] from the project and create a new empty file named [App\_q2.cpp] into the project to start your Animal Card Game implementation.  **Sample Run 1:**   |  | | --- | | Enter Player 1's cards: S M E  Enter Player 2's cards: E M S  Player 1 wins. |   **Sample Run 2:**   |  | | --- | | Enter Player 1's cards: E M S  Enter Player 2's cards: E M S  Tie. |   **Sample Run 3:**   |  | | --- | | Enter Player 1's cards: E E M  Enter Player 2's cards: S M S  Player 2 wins. |   **Question for you to ponder:**  Will the Animal Card Game run forever? Explain your answer.  Note that, the user input as shown below is not allowed because the game only consists of 2 Mouse, 2 Snake, and 2 Elephant cards.  Enter Player 1's cards: E M S  Enter Player 2's cards: S S E |