Common mistakes for Memory Game Project.

Error

```
./run_autograder: line 10:    20 Killed                  python run_tests.py
> /autograder/results/results.json

Your results.json file could not be parsed as JSON. Its contents are as
follows:
```

Possible reasons:

(1) Code is not compiled.
(2) Set all elements of bShown array to be true in play method. Then you cannot find an index i that is in [0, numSlots) and bShown[i] is false.
(3) Endless loop or codes surpass time limit.

Work order:

(1) Define swap function.
(2) Define randomize function.
(3) Submit your current code to autograder. You should get 20 out 100. If not, fix functions swap and randomize first, **afterwards**, work on constructor.
    (a) **Comment** srand(time(NULL)); otherwise, the code generates random output and cannot be graded by autograder.
    (b) Generate numPairs random integers in [0, 1000), that is, there is at most 3 digits in generated the integers. Note numPairs random integers must be generated **before** calling randomize function, otherwise, your code may work, but the numbers generated will be different from that of autograder and cannot get grades.
    (c) Call randomize function.
    (d) Use the results in (b) and (c) to put numPairs random integers in a string array of size numSlots.
    (e) Do not forget to release dynamically allocated memory in (b) – if you use dynamically allocated memory array -- and (c).
(4) Work on destructor. Release dynamically allocated memory of data member values and handle dangling pointer problem.
(5) Work on display method. Note that the format needs to be matched exactly, otherwise, you cannot work play method, which uses display to show results in each step.
    To compare two text file output, you can google “text comparison”. For example, use https://text-compare.com/. Or you can use diff command in Mac/Linux.
(6) Work on play method.
    Keys:
    (a) Declare a bool array bShown of size numSlots. The value of bShown[i] decides the corresponding item in data member values – that is, values[i] -- is shown or not, for 0 <= i and i < numSlots.

(b) Initialize each element of bShown to be **false**. If you set each element to be true, then due to step (c), your code will be in endless loop since you cannot find an index that is not shown yet.
(c) Declare numFlips and pairsFound to be int and initialize them to be zero. Declare variable toBeMatched to be an int, representing the index of the first card in each round. Variable toBeMatched will be initialized later.
(d) Make sure input a valid index that is not shown yet. A valid index means the index must be in the range of [0, numSlots). That is, index >= 0 && index < numSlots. An index whose element in data member values is not shown yet means that bShown[index] == false. Any index not satisfy the above condition needs to be re-entered. A pseudo code is as follows.

```
Enter an index
while (index < 0 || index >= numSlots || bShown[index] == true)
begin
    prompt which error is, can be EITHER index out of range OR the index is shown
    prompt re-enter an index
    re-enter index
end
```

Warning: you cannot break the above repetition statement as

```
while (index < 0 || index >= numSlots)
    …

while (bShown[index] == true) //index is not checked before you use it.
    …
```

(e) Each time a valid not-shown-yet index is chose, numFlips is increased by 1.
(f) If numFlips is odd, this means the first flip of a round, you need to set the corresponding index of bShown to be true – that is, shown the first card flipped – and remember the index to which card is flipped in variable toBeMatched.
(g) Otherwise, numflips must be even, check the current index with toBeMatched, if both elements in values matched and they are not empty string, show the current card (ie, the second flip of this round), increase variable pairsFound to be true, otherwise, unshown the first card in this round, that is, set bShown[toBeMatched] to be false.
(h) Display the current layout of values.
(i) Repeat Steps (d) – (h) until all non-empty matched pairs are found, a total of numPairs.
(j) Report total number of flips as “Congratulations! Take … steps to find all matched pairs.”. **Note that this prompt is ended with a new line character**.

A complete pseudo code for play method is shown as follows.

```
Set bShown to be an array of bool type with size numSlots. Each element is initialized to be false.
Declare and initialize numFlips to be 0.
Declare and initialize pairsFound to be 0.
Declare toBeMatched to be int.

while (not all pairs are found) //hint: what is the relation with numPairs and pairsFound to indicate not all pairs are found
begin
    //choose a valid index such that 0 <= index < numSlots and bShown[index] is false, ie, values[index] is not shown yet.
    //hint: use a while loop in Step (d)
    Enter an index
    while (index < 0 || index >= numSlots || bShown[index] == true)
    begin
        prompt which error is, can be EITHER index out of range OR the index is shown
        prompt re-enter an index
        re-enter index
    end

    numFlips++;

    if (numFlips is odd) //first card of a round, a round involves flipping of two cards
    begin
        show the current card //hint: bShown[index] is set to what value?
        remember the card to be matched //hint: toBeMatched = ?
    end
    else //second card of a round, need to check whether the current index is matched with toBeMatched or not
    begin
        if (....) //you fill in the condition to means what means a match, compare values[index] and values[toBeMatched] and they are not empty-string cards
        begin
          pairsFound++
          show the second card in this round, ie, set bShown[index] to be ...
        end
        else
        begin
            unflip the first card in this round, ie, set bShown[toBeMatched] to be ....
        end
    end
```

display the current layout by calling display method
end

Print “Congratulations! Take … steps to find all matched pairs.” with a new line character.

- After implementing the above codes, you should get 32 steps for the specific exam tested by autograder. If not, check play method.
- If the number of steps matches, and you still do not get grade for play method, do a text comparison of your output and the expected output.