Worksheet 1: Classes and Pointers

Problem 1

For each of the following parts, write a single C++ statement that performs the indicated task. For each part, assume that all previous statements have been executed (e.g., when doing part e, assume the statements you wrote for parts a through d have been executed).

a.) Declare a pointer variable named fp that can point to a variable of type string.

b.) Declare fish to be a 5-element array of strings.

c.) Make the fp variable point to the last element of fish.

d.) Make the string pointed to by fp equal to "salmon", using the * operator.

e.) Without using the fp pointer, and without using square brackets, set the element at index 3 of the fish array to have the value "yellowtail".

f.) Move the fp pointer back by three strings.

g.) Using square brackets, but without using the name fish, set the element at index 2 of the fish array to have the value "eel".

h.) Without using the * operator, but using square backets, set the string pointed to by fp to have the value "tuna".

i.) Declare a bool variable named d and initialize it with an expression that evaluates to true if fp points to the string at the start of the fish array, and false otherwise.

j.) Using the * operator in the initialization expression, declare a bool variable named b and initialize it to true if the string pointed to by fp is equal to the string immediately following the string pointed to by fp, and false otherwise.

Problem 2

Assume the following variable declarations:

```
int bird = 0;
int *binoculars = &bird;
```

Which of the following statements change the value of bird to 6?

```
a.)  binoculars = 6;
b.)  bird = 6;
c.)  (*bird) = 6;
d.)  (*binoculars) = 6;
e.)  a. and b. only
f.)  a. and d. only
g.)  b. and d. only
h.)  c. and d. only
```

Problem 3

Suppose you're tasked with fixing a function definition that does not work as intended. The function is supposed to compare two strings and set the count to the number of identical characters, two characters are identical if they are the same character and are in the same position in the cstring. Note that cstrings are just character arrays that have '\0' as their last character, for example

```
     char name[7] = "harry";
```

might looks like this in memory:

h	**a**	**r**	**r**	**y**	**\0**	

The function usage is as follows:

```
compareCstrings("SMC", "SBCC", count); // should set count to 2
compareCstrings("basketball", "Baseball", count); // should set count to 2
```

Currently the function definition is:

```
void compareCstrings(const char *str1, const char *str2, int &count) {
     *count  = 0;
     while (str1 != '\0' || str2 != '\0') {
          if( *str1 == *str2)
               *count++;
          str1++;
          str2++;
     }
}
```

But this does not work, determine why not?

Problem 4

Implement the class Hotel, as declared below, which keeps track of the reservation status of each room in a hotel. Each room can be RESERVED, OCCUPIED, or EMPTY, and this information is stored in a 2-dimensional array, where each row represents a floor, and each column represents a room. Each room is represented by an integer (e.g., 425). For example, the status of room 425 is stored in m_rooms[4][25].

```
const char RESERVED = 'R';
const char OCCUPIED = 'O';
const char EMPTY = 'E';
const int FLOORS = 20;

const int ROOMSPERFLOOR = 50;

class Hotel
{
public:
     Hotel();
     bool reserve(int roomNum);
     bool cancel(int roomNum);
     bool checkIn(int roomNum);
     bool checkOut(int roomNum);
     int numEmpty(int floor) const;

private:
     char m_rooms[FLOORS][ROOMSPERFLOOR];
     // More private members here, if necessary.



};

Hotel::Hotel()
{
     // EMPTY the rooms.
     for (int i = 0; i < FLOORS; i++)
          for (int j = 0; j < ROOMSPERFLOOR; j++)
               m_rooms[i][j] = EMPTY;
}
```

Implement other functions below.

```
bool Hotel::reserve(int roomNum)
{
     // TODO: If the room is EMPTY, set it to RESERVED, and return true.
     // In all other cases, do not change anything and return false.






}
```

```
bool Hotel::cancel(int roomNum)
{
      // TODO: If the room is RESERVED, set it to EMPTY, and return true.
      // In all other cases, do not change anything and return false.






}

bool Hotel::checkIn(int roomNum)
{
      // TODO: If the room is RESERVED, set it to OCCUPIED, and return true.
      // In all other cases, do not change anything and return false.






}

bool Hotel::checkOut(int roomNum)
{
      // TODO: If the room is OCCUPIED, set it to EMPTY, and return true.
      // In all other cases, do not change anything and return false.






}

int Hotel::numEmpty(int floor)
{
      // TODO: Return the number of empty rooms on the floor.
      // Return -1 if floor is invalid.






}

// Write helper functions down here if necessary.
```

Problem 5

Design `BankAccount` class, which lets the account owner to deposit and withdraw money using a password. The class you write should meet the following requirements:

1. There should be no default constructor.
2. There should be only one constructor, which takes in the initial balance in dollars, and the password. The password is an integer, most likely 4 digits, but we won't enforce that limit here (which means it can even be negative). The initial amount, however, must not be negative, if it is negative, set it to $0.00 by default. Your `BankAccount` class should be able to keep track of the balance up to the last cent.
3. It should support two operations, deposit and withdraw. Both must be Boolean functions. One must specify the amount of money he wants to deposit/withdraw, and the password. If the password is incorrect, the return value must be false. Also, for withdraw, if the requested amount exceeds the current balance or is negative, return false and do not change anything. If the deposit amount is negative, do not change the balance and return false.
4. Add a Boolean function `setPassword`, which takes in two passwords – the original password and the new password. If the original password does not match, return false and do not change the password. If the original password is correct, update the password with the new password.
5. Provide an accessor function balance, which accepts the password. If the password is correct, return the balance. If not, return -1.
6. You can create private member functions and variables as you wish.

A possible usage of this class looks like this:

```
BankAccount ba(500, 1234);     // initial amount: $500, password: 1234
cout << "$"     <<   ba.balance(1234) << endl;  // prints $500.00
cout << "$"     <<   ba.balance(2345) << endl;  // prints $-1
ba.deposit(25,  1234);                          // deposit $25
cout << "$"     <<   ba.balance(1234) << endl;  // prints $525.00
ba.withdraw(500, 2345);                         // should return false
ba.withdraw(1000, 1234);                        // should return false
ba.withdraw(100, 1234);                         // make the withdraw
cout << "$"     <<   ba.balance(1234) << endl;  // prints $425.00
ba.setPassword(1234, 2345);                     // change the password
ba.withdraw(300, 2345);                         // make the withdraw
```

Worksheet 1: Classes and Pointers

(a) Write the full class declaration here.

(b) Write the definition (implementation) of the constructor, including the header.

(c) Write the definition of balance, including the header.

(d) Write the definitions of deposit and withdraw, including the header.

(e) Write the definition of setPassword, including the header.

(f) If you used any extra private member function(s), provide the definition(s) here.