# F18 CS20A Assignment 1

## STEWART DULANEY

**TOTAL POINTS** 

## 35.5 / 36

QUESTION 1

10 pts

1.1 1/1

√ - 0 pts Correct: False (Underscore)

- 1 pts Incorrect: True

1.2 1/1

√ - 0 pts Correct: False

- 1 pts Incorrect: True

1.3 1/1

√ - 0 pts Correct: True

- 1 pts Incorrect: False

1.4 1/1

√ - 0 pts Correct: False

- 1 pts Incorrect: True

1.5 1/1

√ - 0 pts Correct: False

- 1 pts Incorrect: True

1.6 1/1

√ - 0 pts Correct: True

- 1 pts Incorrect: False

1.7 1/1

√ - 0 pts Correct: False

- 1 pts Incorrect: True

1.8 1/1

√ - 0 pts Correct: True

- 1 pts Incorrect: False

1.9 1/1

√ - 0 pts Correct: False

- 1 pts Incorrect: True

1.10 1/1

√ - 0 pts Correct: True

- 1 pts Incorrect: False

**QUESTION 2** 

2 4/4

√ - 0 pts Correct

- 1 pts reference or pointer count

- 1 pts && logic

**- 1 pts** index = 0;

- 1 pts str1[index], str2[index]

QUESTION 3

10 pts

3.1 2 / 2

√ - 0 pts Correct

- 0.5 pts Mostly correct

- 1 pts Missing some points

- 2 pts Missing

3.2 2/2

√ - 0 pts Correct

- 1 pts Overloading

- 1 pts Different types or number of formal

parameters

- 2 pts Missing

3.3 2/2

√ - 0 pts Correct

- 1 pts Partially Correct

- 2 pts Incorrect/Missing

3.4 2/2

√ - 0 pts Correct

- 1 pts Partially Correct

- 1.5 pts Mostly incorrect

- 2 pts Incorrect/Missing

3.5 2/2

√ - 0 pts Correct

- **0.5 pts** Mostly Correct

- 0.5 pts Inconsistency between numbers and

output

- 1 pts Partially Correct

- 1.5 pts Incomplete
- 2 pts Incorrect/Missing

# QUESTION 4

# 12 pts

# 4.1 Constructor 2 / 2

## √ - 0 pts Correct

- **0.5 pts** Minor syntax
- 1 pts logic error
- 2 pts Missing/Incorrect

#### 4.2 reserve 2 / 2

## √ - 0 pts Correct

- 0.5 pts Minor logic error
- **0.5 pts** Syntax
- 1 pts Logic error
- 1.5 pts Largely incorrect
- 2 pts Incorrect/Missing

#### 4.3 cancel 2 / 2

#### √ - 0 pts Correct

- 0.5 pts Syntax
- **5 pts** Minor logic
- 1 pts Logic error
- 1.5 pts Largely incorrect
- 2 pts Incorrect/Missing

# 4.4 chechln 2 / 2

# √ - 0 pts Correct

- 0.5 pts Syntax
- 0.5 pts Minor logic
- 1 pts Logic Error
- 1.5 pts Largely incorrect
- 2 pts Missing/Incorrect

# 4.5 checkOut 2/2

# √ - 0 pts Correct

- 0.5 pts Syntax
- **0.5 pts** Minor logic
- 1 pts Logic Error
- 1.5 pts Largely incorrect
- 2 pts Incorrect/Missing

# 4.6 numEmpty 1.5 / 2

- **0 pts** Correct
- √ 0.5 pts Minor logic

- 1 pts Logic
- 2 pts Incorrect/Missing

Assignment 1

ID: 1545566

# True/False: Circle one

True False Variable and functions identifiers can only begin with alphabet and digit.

2. True / False Array sizes can be non-constant variables.

3. True / False Array sizes must be known at compile time.

4. True / False An array can have size of 0.

5. True / False int m = 15.6; won't compile because the types do not match.

6. True False int n = 11/5; will allocate memory for n and initialize with the value 2.

7. True / False Array indices begin with the number 1.

8. True / False You can define multiple functions with the same identifier.

9. True / False Variables declared local to a function are still accessible after the function completes execution.

10. True / False structs by default have private access specification.

11. 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 <u>same character</u> and are in the <u>same position</u> in the cstring. This function will be case sensitive so the character 'a' is not the same as 'A'. 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	У	\0	

Usage of this function might look like:

Currently the function definition is:

Identify the errors in the above implementation and rewrite the function so that it satisfies specification. Try to keep the general form of the original code, you should not have to add or remove any lines of code, just modify the existing ones.

Assignment 1 ID: 1545566

12. Use the code below to answer the questions that follow. Assume that all proper libraries and name spaces are included and that the code will compile without error. Within the main function, for the variable int last\_sid, write in the last digit of your SMC student ID.

```
int foo(int a, int b) { //First
   int c = a+b;
   while(c > = 3)
      c-=3;
   return c;
char foo(string a, int b) { //Second
   return a[b];
string foo(int b, string &a) { //Third
   string sub = a.substr(3*b,3);
   a.replace(3*b,3,"...");
   return sub;
void main() {
   int last_sid = <u>6</u>; //<-Last digit of your SID
   string letters("ggfiorkcboneat !!!ws adtarojot");
   string output("");
   int numbers[] = \{0,8,3,7,4,6,9,1,2,5\}; // size 10
   for(int i=0; i<10; i++) {
       int j = numbers[i];
                                        // first
       numbers[i] = foo(last sid,i);
       string s = foo(j, letters);
                                       11 third
       output += foo(s, numbers[i]);
                                       11 second
   cout << output;
}
```

- **a.)** For each of the three foo functions *briefly describe in plain language* what each function is doing. You may refer to the top function as the first function, the one below as the second function, and finally the last as the third function.
- The first function computes and returns the integer (a + b) mod 3.
- The second function returns the character at index b in the string parameter a.
- The third function returns the string equal to the substring from index (3\*6) to (3\*6 + 3) of string parameter a. It also modifies the reference string parameter a by replacing the substring returned with the string literal "..." in the original
  - b.) What is it called when we use the same function identifier for multiple functions? What must we done to allow the compiler to differentiate between functions with the same identifier?

It is called function overloading. Each overloaded function must have a different function signature, namely, it must have a different number of parameters or its' parameters must have different types so the compiler knows which function to call.

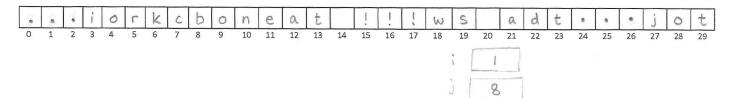
Santa Monica College Fall 2018 CS 20A: Data Structures with C++

Name: Stewart Dulaney

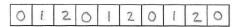
Assignment 1

ID: 1545566

c.) What does the string letters look like at the end of second iteration? The blocks below are placed for your convenience, you may overwrite the values.

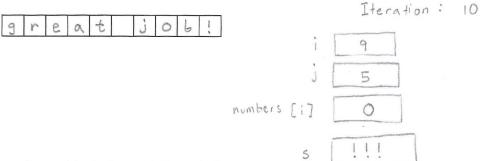


d.) What does the numbers array look like at the end of the program? The blocks below are placed for your convenience, you may overwrite the values.

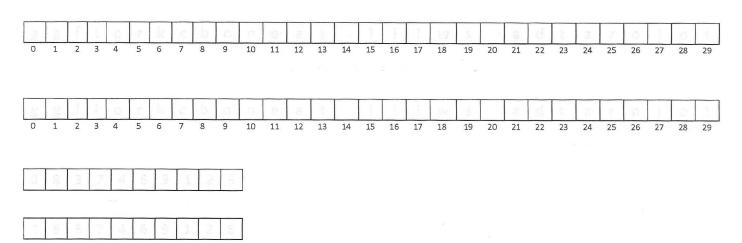




f.) (3 points) What is printed with cout << output << endl; ?



\*Extra blocks for work if needed:



Name: Stewart Dulaney

Assignment 1 ID: 1545566

13. 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]. Note that this hotel has a room 0 on floor 0. For this problem you may not define any additional member variables or functions.

Below is the definition of the Hotel class:

```
// In hotel.h
   const char RESERVED = 'R';
   const char OCCUPIED = '0';
   const char EMPTY = 'E';
   const int FLOORS = 20;
   const int ROOMSPERFLOOR = 50;
   class Hotel {
   public:
         Hotel();
                                         // TODO
         bool reserve(int roomNum);
                                         // TODO
                                         // TODO
         bool cancel(int roomNum);
         bool checkIn(int roomNum);
                                         // TODO
         bool checkOut(int roomNum);
                                         // TODO
         int numEmpty(int floor) const;
                                         // TODO
   private:
         char m_rooms[FLOORS][ROOMSPERFLOOR];
   };
Implement Hotel's functions below:
   // In hotel.cpp
  Hotel::Hotel() {
         // TODO: Set all the rooms in the Hotel to be EMPTY
        for (int i = 0; i < FLOORS; i++) {
              for (int j = 0; j < ROOMSPERFLOOR; j++ ) {
                     m_rooms[i][j] = EMPTY;
  }
```

Name: Stewart Dulaney

ID: 1545566

```
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.
    if ( m - rooms [ room Num / 100] [ room Num % 100] == EMPTY ) {
          m-rooms [room Num / 100][ room Num % 100] = RESERVED;
          return true;
    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.
    if (m-rooms froom Num / 100] room Num % 100] == RESERVED) {
       m-rooms [roomNum / 100] [roomNum % 100] = EMPTY;
       return true;
  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.
   if (m_rooms [roomNum / 100][roomNum % 100] == RESERVED) {
       m_rooms [room Num / 100] [room Num % 100] = OCCUPIED;
       return true;
  return false;
}
```

Name: Stewart Dulaney

Assignment 1 ID: 1545566

```
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.
   if (m_rooms [room Num / 100] [room Num % 100] == OCCUPIED) {
       M-rooms [room Num / 100] [room Num % 100] = EMPTY;
      return true;
  return false:
int Hotel::numEmpty(int floor)^{
     // TODO: Return the number of empty rooms on the floor.
     // Return -1 if floor is invalid.
                             (floor > FLOORS)) {
     if ((floor < 0)
          return -1:
    int count = 0;
    for (int i = 0; i < ROOMSPERFLOOR; i++) {
        if ( m_rooms [floor][i] == EMPTY ) {
           count ++;
  return count;
```