START EARLY!  This program will use a one-dimensional array and functions to manage motel room reservations, display occupancy, sort elements, lowest and highest number of guests in a room. The six rooms in the motel each have a queen bed and 2 twin beds for a maximum of 4 guests per room.

a) To increase readability, use symbolic constants e.g. " #define NUM_ROOM 6 "

b) You will write 4 functions besides main(). main() will call (invoke) all functions except swap(). The function declarations are:

```c
void reserve (int a[]);          // Reserve a room for guests
int occupancy (int *a);           // Display room occupancy
void bubbleSort(int a[]);         // Ascending sort
void swap (int *p1, int *p2);     // Swap elements
```

Array notation is used in functions reserve() and bubble_sort(). Pointer notation is used in swap() and occupancy().

c) main() initializes an array, "int cMotel[NUM_ROOM] = {1, 4, 2, 2, 3, 3};"
e) **reserve()** uses array notation to read user input into a temporary variable, `num`. Test if `num` is within the valid range of zero through 4 guests per room.

If valid, then ASSIGN into array, else prompt again for ONLY the current input value to be typed in again. Repeat input and test until a valid value is entered within the for loop.

```c
int num;                        // Temporary input number
for(...) {                     // Loop through array elements
    Prompt for Room#          // 1st, 2nd, 3rd, ...
    scanf("%d", &num);       // Read into temporary variable
    getchar();              // Read <ENTER> key
    Error check for number in range of 0 through 4
    true
        Decrement loop counter
        Print error message
    false
        Assign num to a[i]   // Valid input, assign into array
}
```

If `num` is a valid value, then assign into the element of the 1-dimensional array. A for loop, will repeat the prompt until ALL six valid numbers are entered. Do not manipulate array for an input value in incorrect range. Call other functions only upon valid input. See sample output for example.

f) **bubbleSort()** will use a “bubble sort” algorithm by bubbling up the smallest value into first element calling **swap()**. **main()** will print sorted array.

```c
void bubbleSort( int a[] )
{
    int i, j;
    for(i = 0 ; i < NUM_ROOM-1 ; ++i)
        for(j = NUM_ROOM-1 ; i < j ; --j)
            if( a[j-1] > a[j] )
                swap(a[j-1], a[j]);  // Swap 2 elements
}
```

g) **swap()** is same as in Pointer handout except using integers.

h) Your program must loop so that the user can continue to generate and view the output until the user responds with 'n' or 'N' to the offer "Want more motel management? ". NOTE: Your program will end **ONLY** with the input of 'n' or 'N'.

**HINT:** Solve this problem in small steps. Here's a suggestion.
1) In **main()**, write the code for step a), b), c) above. Test.
2) Write the code for step f) above (**bubbleSort()**).
3) Write the code for step g) above (**swap()**).
4) Write the code for step d) above (**reserve()**). Test, then code for error input later.
5) Write the code for step h) above so your entire program will loop again.
6) Complete error checking. **DONE!**
C MOTEL MANAGEMENT
==================

OCCUPANCY: 1 4 2 2 3 3

RESERVATIONS:
Enter ROOM #1: 4
Enter ROOM #2: 3
Enter ROOM #3: 2
Enter ROOM #4: 1
Enter ROOM #5: 4
Enter ROOM #6: 1

OCCUPANCY: 4 3 2 1 4 1
SORTED: 1 1 2 3 4 4
LOWEST: 1
HIGHEST: 4

Want more motel management? X

OCCUPANCY: 1 1 2 3 4 4

RESERVATIONS:
Enter ROOM #1: 4
Enter ROOM #2: 3
Enter ROOM #3: 2
Enter ROOM #4: 1
Enter ROOM #5: 5
ERROR! Enter a number (0-4)!
Enter ROOM #5: 6
ERROR! Enter a number (0-4)!
Enter ROOM #5: 4
Enter ROOM #6: -1
ERROR! Enter a number (0-4)!
Enter ROOM #6: 0

OCCUPANCY: 4 3 2 1 4 0
!! VACANCY !!
SORTED: 0 1 2 3 4 4
LOWEST: 0
HIGHEST: 4

Want more motel management? N

Submit the final version of your program as p6.c

Verify you SAVED your work in the Documents - cs5f HOME directory.