CMSC 11, More exercises on functions involving arrays and strings

It's nice to know that some students are posing programming problems among themselves. I hope that means that at least some people are really starting to enjoy problem solving and programming. The list below of programming problems is again somewhat graduated, starting with (what I think are) the easier problems first. For your solutions, also provide main() drivers that would test your functions.

```
int stringlength( char *s )
// a do-it-yourself version without using the strlen() function

char * stringcopy( char *dest, char *source )
// a do-it-yourself version without using the strcpy() function

int search( int n, int a[ ], int target )
// search for the target in the array a[] of n integers;
// if the target integer is present in the array, return the position of its first occurence,
// otherwise return the special code -1 to indicate its absence

int sorted_test( int n,  int a[ ] )
// test the status of a given array a[] of n integers; return 0 if it is unsorted,
// return 1 if the items are in ascending order, -1 if the items are in descending order

int sort( int n,  int a[ ] )
// arrange the array a[] in ascending order by swapping pairs that are out of order
// also return the total number of swaps performed in the process

int remove_all_blanks( char *s  )
// compress the string s by removing all blanks, and return the length of the new string
// e.g., the string “compress    this    string” will be transformed into the string “compressthisstring”

char palindrome_test( char *s )
// checks if the string s reads the same forwards and backwards (return 1 if yes, 0 otherwise)
// you might also want to consider case-insensitivity, e.g., “Madam”, or even phrases with blanks and
// punctuation such as “Madam I'm Adam”

void shuffle( int n,  int a[ ] )
// given an array of n integers, this procedure will randomly shuffle the array; the same numbers
// will form the array but in random positions, e.g., { 1, 2, 3, 1 } might change to { 3, 1, 1, 2 }

int remove_duplicate_chars( char *s )
// remove all adjacent duplicate characters in the string s, and return the length of the new string,
// e.g.,  “bookkeeper--ssssh” is transformed into “bokeper-sh” (return the new length 10)

void report_aliens( int n, char *class[], int m, char *subscribers[] )
// given alphabetically-sorted class lists and subscriber lists, of sizes n and m, respectively,
// print the names of those subscribers that are not in the class list

char * decimal_to_roman( int n, char *roman )
// given a positive integer n, convert it into Roman, e.g., 34 will be converted to 'XXXIV'

int roman_to_decimal( char *roman )
// given a string of symbols from the Roman numeric alphabet { I, V, X, L, C, D, M }, evaluate and
// convert it to decimal; if the string is invalid return a special code such as -1 that indicates a syntax
// error, e.g., “MCMXC” will be evaluated as 1990
```

jmsamaniego@uplb.edu.ph