CSE108

LAB MIDTERM – SPRING 2015

PART 1 (20 pts)

```
int find_indice(const int arr[], int size, int target, int *ind_1, int *ind_2);
```

Write a function that finds indices of two elements in an array whose sum equals to a target value. For example, if the array is given as {0,3,6,2,-5} and target value is given as 1, function will assign 2 to first index and 4 to second index and **return 1**. Note that indices are return values of that function too. **If exact match is not found**, function will **assign -1 to indices and return 0**. After implementation you are expected to call that function in main and print result to console as given format below. **Use array and target values below** to test your implementation and check your result. If match is not found print not found to console. You must get target value from console input.

		Test 1	**Test 2**
FUNCTION INPUT	**Array**	{0,1,3,4,5,6,7,8,9,15,-5,0,9,10,23}	
	Target	-2	-20
FUNCTION OUTPUT	**First Indice**	2	-1
	Second Indice	10	-1
PRINTED MESSAGE TO CONSOLE	**Print Format**	array[2] => 3, array[10] => -5	Not found!

PART 2 (25 pts)

```
Bool is_sorted(const char *arr);
```

Write a **recursive** function '**is_sorted**' that takes a string as an input parameter and returns **TRUE** if characters in the string are in alphabetical ascending order; returns **FALSE** otherwise. **Hint:** The characters may be **lower-case** or **upper-case**. Write a driver to test your function for the follow examples.

Example: Assume **example_array1** is **"AstyOlpndWsAfae", is_sorted(example_array1)** will return **FALSE** and assume **example_array2** is **"aGjklOvz", is_sorted(example_array2)** will return **TRUE.**

PART 3 (20 pts) Write a function "**is_element**" which determines if **a character** exists in **a string array** (2 dimensional array of characters). Your function should take a character and a string array as input parameters and return "1" if the character exists in any line of the array and "0" otherwise. You should use the "**strchr**" function of the string library in your implementation. (`char* strchr(const char* s, char c);`) Function returns the address of the first occurrence of c in s if c exists in s, and NULL pointer otherwise. Write a driver to test your function for the examples given below.

String Array	Character	Return Value
{"abc", "sbc", dsds}	s	1
{"abc", "sbc", dsds}	t	0

PART 4 (35 pts)

```
void fill(int matrix[MAX_SIZE][MAX_SIZE], int n);
void print_mat(const int matrix[MAX_SIZE][MAX_SIZE], int n);
```

In this part, you will write two functions above to produce a spiral matrix and print it. A spiral array is a square arrangement of the first N*N natural numbers, where the numbers will start with 0 and increase sequentially as you go around the edges of the array spiraling inwards. It follows the left, down, right and upward directions, respectively. Use the function "**fill**" to fill the given matrix and the function "**print_mat**" to print the matrix with the prototype above. n is the real size of the matrix n*n. Write a driver to test your functions with the sizes 10 and 9. For example;

```
Size of the matrix : 9
 0  1  2  3  4  5  6  7  8
31 32 33 34 35 36 37 38  9
30 55 56 57 58 59 60 39 10
29 54 71 72 73 74 61 40 11
28 53 70 79 80 75 62 41 12
27 52 69 78 77 76 63 42 13
26 51 68 67 66 65 64 43 14
25 50 49 48 47 46 45 44 15
24 23 22 21 20 19 18 17 16
```

```
Size of the matrix : 10
 0  1  2  3  4  5  6  7  8  9
35 36 37 38 39 40 41 42 43 10
34 63 64 65 66 67 68 69 44 11
33 62 83 84 85 86 87 70 45 12
32 61 82 95 96 97 88 71 46 13
31 60 81 94 99 98 89 72 47 14
30 59 80 93 92 91 90 73 48 15
29 58 79 78 77 76 75 74 49 16
28 57 56 55 54 53 52 51 50 17
27 26 25 24 23 22 21 20 19 18
```

PART 5 (25 pts) You need to implement the algorithm based on the example given in the table. Let n and m be positive integers whose product we want to compute. The idea is to take half of the first number and double the second number repeatedly till the first number becomes 1. In the process, whenever the first number become odd, we add the second number to result (result is initialized as 0). Write a recursive function which takes n and m as input parameters and returns nxm using the algorithm above. Write a driver to test your function.

n	**m**	
50	65	
25	130	130
12	260	
6	520	
3	1040	1040
1	2080	2080
		+----------
		3250