COEN 11- Homework 2
Solutions

1. What will this code output (printf)?

```
int main (void)
{
        int i;
        int x[5]={0,1,2,3,4};
        int *p=x;
        int a=0;
        int b=10;

        x[0] += f (a, &b, x);
        printf ("%d, %d\n", a, b);
        for (i = 0; i < 5; i++)
                printf ("%d\n", x[i]);
        return 0;
}

int f (int x, int *y, int *z)
{
        x += 100;
        *y += 200;
        z[2] += 300;
        return (x);
}
```

Output

```
0, 210
100
1
302
3
4
```

From Homework 1, rewrite the solutions for the problems 2, 3, 4 and 5 using pointers.

2. Write a function to return the sum of all the elements in a 2D array of size NROWSxNCOLS.
   The prototype of the function is: int sum (int [][NCOLS]);

```
int sum (int x[][NCOLS])
{
        int i, size =  NROWS * NCOLS;
        int *p = x[0];
        for (i=0; i < size; i++, p++)
                sum += *p;
        return sum;
}
```

3. Write a function to initialize 2D array x (size MxM) with the following pattern (shown for a 5x5 array):

```
1 0 0 0 1
0 1 0 1 0
0 0 1 0 0
0 1 0 1 0
1 0 0 0 1
```

```
void init (void)
{
	int i, j, *p;
	for (i=0; i<M; i++)
	{
		p=x[i];
		for (j=0; j<M; j++, p++)
			if (i==j ||i+j == M-1)
				*p = 1;
			else
				*p = 0;
	}
	return;
}
```

4. Write a function to return the number of sub-strings (sequence of characters with no spaces, tabs, or newlines) in string str received as argument. The prototype of the function is:
int count_strings (char *);

```
int count_strings(char *str)
{
	int flag=0, int counter = 0;
	while (*str != '\0')
	{
		if (*str != ' ' && *str != '\t' && *str != '\n')
		{
			if (flag == 0)
			{
				counter++;
				flag = 1;
			}
		}
		else
			flag = 0;
		str++;
	} //end while
	return counter;
}
```

5. Write a function to return the length of the longest string in an array of strings (size NROWSxNCOLS) received as argument. Do not use strlen! The prototype of the function is: int largest_size (char [][NCOLS]);

```
int largest_size (char strings[][NCOLS])
{
        int largest = 0, size, i;
        char *p;
        for (i=0; i < NROWS; i++)
        {
                size=0;
                p=strings[i];
                while (*p != '\0')
                {
                        size++;
                        p++;
                }
                if (size > largest)
                        largest = size;
        }
        return largest;
}
```