**Lab 12**

**Pointers and Dynamic Memory Management**

1. Generates a two dimensional matrix, and fills it randomly withzeroes and ones. (Use pointers, allocate memory).
2. Write a program that start with reading an unknown number of values into dynamically allocated array and print the numbers of values that were read, main function should print on separate lines:
   1. The number of values read
   2. The minimum value
   3. The maximum value
   4. The average of the values;

**Use Pointersnot Array!**

1. Read about strtok:

**char \*strtok (s1, s2): Split string into tokens**

s1

C string to truncate.  
Notice that this string is modified by being broken into smaller strings (tokens).  
Alternativelly, a null pointer may be specified, in which case the function continues scanning where a previous successful call to the function ended.

S2

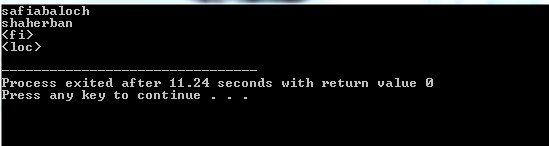
C string containing the delimiter characters.  
These can be different from one call to another.

**Return Value**

If a token is found, a pointer to the beginning of the token.  
Otherwise, a *null pointer*.  
A *null pointer* is always returned when the end of the string (i.e., a null character) is reached in the string being scanned.

Write a program which takes two separate strings and find out the delimiters.

For example: string1 = Safiabaloch, string2 = shaherban; the output is:



1. What is wrong with the following code.
   1. #include <stdio.h>

#include <stdlib.h>

#include <string.h>

intmain(intargc, char\* argv[]){

char\* name = malloc(strlen(argv[1])) ;

name = strcpy(name,argv[1]);

printf("%s \n", name);

return EXIT\_SUCCESS;}

}

* 1. #include <stdio.h>

#include <stdlib.h>

#include <string.h>

intmain(intargc, char\* argv[]){

int\* ptr = malloc(4);

free(ptr);

scanf("%d",\*ptr);

return EXIT\_SUCCESS;}

* 1. intmain(intargc, char\* argv[]){

int\*\* A;

foo(&A);

return 0;

}

foo(int\*\*\* array){

int\*\* arrayint = (int\*\*)malloc(2\*n\*sizeof(int\*));

for (i=0;i<n;i++)

arrayint[i] = (\*array)[i];

free(\*array);

array = &arrayint;

}