CS 223, Fall 2016, Assignment 1
Push to a1 repo by Tuesday, September 13

You are to write a program that will implement the "division algorithm" in the form of a module (method, function, procedure, subroutine, …) called as

Void DivAlg(unsigned int V, unsigned int B);

which will use the division algorithm to a) produce remainders mod B in the range 0 .. B-1, and b) put the character equivalents of the remainders into an array as they are produced (e.g. 4 and '4' are different), then c) output the digits created in the proper order. The maximum number of digits that can be produced is 32, assuming that our int data type is a 32 byte value. Assume that B can be at most 16.

You should simply call DivAlg several times from the main. For each value, you should output the values V and B using the %u format descriptor, and then output the characters your algorithm has generated and stored in some buffer. A buffer is just a holding place (a character array in this case). This can be a char[ ] array with a 'null' (ascii code 0) as the last character so that a %s descriptor can be used as in the example below.

To convert a remainder between 0 and 15 to an appropriate character value use a simple table lookup as in char * CharTable = "01234567890ABCDEF"; Notice that because of the free and easy equivalence of pointers and arrays, CharTable[4] = '4', CharTable[10] = 'A', etc.

Hand in a script of your work that includes a) the source code, and b) the execution of your program. Again, simply call your routine several times with interesting values. Be sure to document your code well, including who, what, when due, etc. You can't comment too much.

Some example code using arrays and pointers, though you may not need this on Assignment 1.

==========

```
#include <stdio.h>
void Swap(char *, char * ); // A letter swapper prototype
// playing around with C strings, arrays, pointers
int main ( )
{ int i,j; // some subscripts for the character array
  char A[6] = {'a','b','c','d','e', (char) 0};
     // same as "abcde" but writeable
  char * Ptr1, *Ptr2;
// Let's reverse the string
i = 0; j=4;


printf("Before  %s \n", A);
   while (i < j) {
```

```
      Swap(& A[i++],& A[j--]);
     } ;
  printf("After %s \n", A);

    Ptr1 = A; // name of array <==> &A[0]
    Ptr2 = &A[4];
    printf("Before  %s \n", A);
    while (Ptr1 < Ptr2) {
      Swap(Ptr1++,Ptr2--);  // N.B. Ptr1++ and Ptr1+1 may not be the same !!
     } ;
  printf("After %s \n", A)
}
void Swap(char *P1, char * P2)
{ char C;
   C = *P1 ; // dereference on the right
   *P1 = *P2 ; // deref on left and right
   *P2 = C;
// the switcheroo
}
```