1. Create a function that returns the k-th minimum value of an array. So for example, calling min\_k(nums, n, 2) should return the 2nd smallest element in the array. Here is the function signature:

int min\_k(int nums[], int nums\_len, int k)

The input to this function is the array of integers, the length of the array, and an integer k specifying which minimum value to find. For instance if k=3 you should find the third smallest element in nums. If there are duplicate elements, you should only count them once.

1. Write a program that fills an array A of size 5 filled with random integers in the range 1 to 100 and prints it out (for testing purposes). Write programs that then after creating such a random array, do each of the following:  
   (a) Calculate the sum of the elements of A and print it out.  
   (b) Count the number of elements in A whose contents are less than 10.  
   (c) Create a new array B of size 5 such that for all B[i] (0<=i<=19), B[i] = A[0] + A[1] + ... + A[i]

Modify your code to have an arrays of size 20 (so that you are filling an array A of size 20 with random integers...). If you cannot do this with a single edit, modify your program so that you can.

1. Write a program for printing the largest elements in each row of an array using function. The function should take an array and number of rows and columns as parameters.
2. Search an element in an array where difference between adjacent elements is 1. Given an array where difference between adjacent elements is 1, write C program search for an element in the array and return the position of the element (return the first occurrence).
3. Write a C program to find the smallest positive integer value that cannot be represented as sum of any subset of a given array?
4. Write a program to read in a series of text strings. When you get an empty line, the program should return the count of how many times each of the letters (A...Z) and (a...z) appear in the text and then exit, as illustrated below. For this assignment, you may ignore any characters that are not letters, such as numbers or whitespace. You should also count capital and lower-case letters together; for instance, the string "Aaa" should increment the count of 'A' by 3.

Note that char's here are ASCII characters, which are also encoded as numbers from 0..255. Take a quick look at the table of ASCII values [here](http://www.asciitable.com/). For instance, 'A' is value 65 and 'Z' is 90; it is valid to write code like the following:

char buf[MAX\_BUF];

if (buf[0] >= 'A' && buf[0] <= 'Z') /\* ... \*/

in order to figure out if a character in the char array is a capital letter or not. Also note that upper and lower case ASCII values all differ by the fixed amount of 32.

We provided you with some starter code in [ex4.c](https://www.cs.unc.edu/~porter/courses/comp411/s18/lab3/ex4.c). You may assume that no input line will be larger than MAX\_BUF. As with exercise 1, note that fgets still returns one character for the newline, so take this into account when terminating the do loop in the example code.

Some example outputs follow:

% ./ex4

aaaaaaa

Distribution of letters in corpus:

A: 7

B: 0

C: 0

D: 0

E: 0

F: 0

G: 0

H: 0

I: 0

J: 0

K: 0

L: 0

M: 0

N: 0

O: 0

P: 0

Q: 0

R: 0

S: 0

T: 0

U: 0

V: 0

W: 0

X: 0

Y: 0

Z: 0

% ./ex4

asdf;lkj

sdkeiflan

lsweinvk

Distribution of letters in corpus:

A: 2

B: 0

C: 0

D: 2

E: 2

F: 2

G: 0

H: 0

I: 2

J: 1

K: 3

L: 3

M: 0

N: 2

O: 0

P: 0

Q: 0

R: 0

S: 3

T: 0

U: 0

V: 1

W: 1

X: 0

Y: 0

Z: 0

% ./ex4

the quick brown fox jumped over the lazy dog

Distribution of letters in corpus:

A: 1

B: 1

C: 1

D: 2

E: 4

F: 1

G: 1

H: 2

I: 1

J: 1

K: 1

L: 1

M: 1

N: 1

O: 4

P: 1

Q: 1

R: 2

S: 0

T: 2

U: 2

V: 1

W: 1

X: 1

Y: 1

Z: 1