Some notes on using arrays and loops in C

You will have to use loops and arrays in almost every program you will do in the lab from now on, including nested loops. Also, some basic uses of arrays repeat in similar form inside many other programs, like summing, counting etc.

Go through your theory class recordings and slides first. This document is NOT a replacement for them. This just gives some pointers on things that you should learn very well as you have to use them in lab.

In the rest of this document, I will assume that we are working with integers. Change accordingly for float and char. Declare variables etc. also as needed.

Working with arrays and single loops

Before we start: One common mistake while working with arrays:

In the condition inside the for loop, should you start from 0 or 1? Should you go to < or <=? Depends on how you write but mixing them without understanding can be very hard to debug. This is a common mistake we see (even saw many in yesterday's lab), so you should be careful.

If you have an-element array A, remember clearly that the indices in the array are from 0, not 1. Also, the number of elements is n. So to traverse all elements in an array, you can write

```
for (i=0; i < n; i++)
{
    /*Work with A[i] here, so you will go from A[0] to A[n-1] as i changes */
}
```

But the one below is wrong, because when you changed the < to <=, i goes from 0, 1, 2, ...n, i.e., (n+1) terms, but you have only n elements in the array!! C will not give you any error, it will just read some garbage value, and you will get a wrong result, very hard to find what is wrong.

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

What about this below? This one now goes through the correct number of terms (n) as it starts from 1, not 0, so ok there. But if you access A[i] inside the loop, the last value of i will access A[n] (when i=n), but the array elements are only from A[0] to A[n-1]! Also, A[0] will never be accessed. **So again the same problem! If** you do this, inside the loop wherever you wanted to access A[i] earlier, you will have to access A[i-1] (so i=1, will actually access A[0] and so on).

```
for (i=1; i <= n; i++)
```

Suggest you stick with

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

if you want to go through ALL elements of an array A with n elements. But sometimes, we will not want to go through all elements, or not start from index 0, in those cases you will have to change it. Understand what is happening, and you should be able to do this.

Reading in elements in an array A:

You will do this in almost every assignment, so know this by heart.

```
scanf("%d", &n);
for (i=0; i<n; i++)
{
        scanf("%d", &A[i]);
}
```

Printing elements in an array A:

You will do this also in almost every assignment, so know this by heart.

```
for (i=0; i<n; i++)
{
        printf("%d  ", A[i]);
}
printf("\n");
```

Note the space after the %d in the printf, and the extra printf with \n outside the loop. This prints the array elements nicely in one line only separated by spaces, and then prints a newline at the end so that any further print will start in the next line. Run it and see. Remove the space and see what you get. You will remember then.

Finding sum of elements in an array A:

```
sum = 0;
for (i=0; i<n; i++)
{
        sum = sum + A[i];
}
printf("Sum is %d\n", sum);
```

Remember to initialize!!

Counting elements in an array A with some property:

Example: how many elements > 0?

```
count = 0;
for (i=0; i<n; i++)
{
        if (A[i] > 0)
                count = count + 1;
}
printf("Count of elements > 0  is %d\n", count);
```

Remember to initialize!! Also, you could have counted anything else by just changing the condition inside if

Actually, if you look at the above programs, the for loop traverses all elements of the array. So inside the for loop, you can do anything as you consider each element, just print it, sum it, count it (based on some property),…..In different programs that you will write in this lab, you will do different things with them, but keep this basic structure in mind, just change the code inside the loop.

Finding minimum (or maximum):

```
min = A[0];
for (i=1; i<n; i++)
{
        if (A[i] < min)
                min = A[i];
}
printf("Minimum  is %d\n", count);
```

Remember to initialize! Now what will you initialize with? Assume the first one is the min, if not so, your program will change it later. Note that since the first one is already considered, the for loop starts with i=1 and not i=0, so it will check elements A[1], A[2],…..A[n-1].

Question: Can you also remember (in another variable) and print the index in A where the minimum is found? Do this actually and run. If you do not try out actual codes, you will not learn programming by reading only!

Remembering the occurrence of an event with a flag:

Sometimes, you want to check if some property is true or not among the array elements. And if your program finds it true as it looks at each element in the array one by one, you want to

come out of the loop immediately. But then, when you are outside the loop, you need to remember why you came out: is it because (1) the property was found to be true somewhere in the middle of the for loop, **OR** because (2) the property was not found to be true and the loop ended. This was there in your Assignment 7 also, and in the other pdf I sent, I showed you how to do this one way. Here's another way that is used a lot:

Set an integer flag to 0

If you see the property satisfied, set the flag to 1 and break from loop

Check for the flag value outside the loop to distinguish between (1) and (2) above and act accordingly.

Example: You are given an array A of n integers, and another integer k. Does k exists in A (the property you want to check)?

```
flag = 0;
for (i=0; i<n; i++)
{
        if (A[i] == k)
                flag = 1;   /* property is satisfied, so set flag */
}
if (flag == 1)
{
        /* k is found (property is satisfied) */
        printf("The number %d is found in A\n", k);
}
else
{
        printf("The number %d is not found in A\n", k);

}
```

Question: If k is found, can you also remember (in another variable) and print the index in A where it is found?

Working with Arrays and Nested Loops

You need to understand it well. You will have to use it a lot in this lab.

Suppose you have two arrays A of n elements and B of m elements

Do something with A only:

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

```
{
        for (j=0; j<n; j++)
        {
                /* here you have A[i] and A[j]. Can do whatever you want
                   with them */
                /* Just an example below, I want to print all index pairs (i,j)
                    such that A[i] = A[j] */
                if (A[i] == A[j])
                        printf("(%d, %d)  ", i, j);
        }
}
```

Here the outer for loop fixes one value of i, and for that value of i, first all values of j are considered in the inner loop. When the inner loop finishes, the next iteration of the outer loop starts with the next value of i, and so on. So the order in which the elements will be looked at (i value and j value shown as (i, j)) are

In 1st iteration of outer for loop, i =0. Now in inner for loop, j will go from 0 to n-1
(0,0), (0, 1), (0, 2), (0, 3),……and so on……, (0, n-1)

Now inner for loop is over for 1st iteration of outer for loop, and the next iteration of outer for loop starts with i=1
(1,0), (1, 1), (1, 2), (1, 3),……and so on……, (1, n-1)

This continues like
(2,0), (2, 1), (2, 2), (2, 3),……and so on……, (2, n-1)

(3,0), (3, 1), (3, 2), (3, 3),……and so on……, (3, n-1)

And so one till

(n-1, 0), (n-1, 1), (n-1, 2), (n-1, 3),……and so on……, (n-1, n-1)

Remember this order. You can (and should now) print out (i, j) inside the inner for loop to see what gets printed and verify that you understood this order correctly.

Question: A mode of a set is the element that occurs the maximum number of times in the set. Can you count the number of times an element occurs in A, keep track of it, and then print the mode? Do this, this will give you practice with nested loops, counting, and keeping track of maximum together. You know how to do them individually, now combine them, you will have to do things like this in the lab. For example, Assignment 7 could have ben done by counting the number of occurrence of each element in A and printing it out if the count is 1, right?

Do something with A and B:

Same as above, just that the inner loop goes till m (the number of elements in B)

```
for (i=0; i<n; i++)
{
        for (j=0; j<m; j++)
        {
                /* here you have A[i] and B[j]. Can do whatever you want
                   with them */
        }
}
```

Question: Can you write the codes to find and print:

- Set union and set intersection, where A and B are the two sets? (read in the values from the keyboard etc. as usual)