CSC180 tutorial notes – week of Nov. 24

If possible, try to get the students to come to the board and write the solutions, helping the student at the board along the way. I’ve tried it during the extra tutorial sessions, and it works about half the time...

Write a function that prints out an array of integers given the array and its size without using loops

```c
void print_arr(int a[], int size)
{
    if(size == 0)
        return;
    printf("%d ", a[0]);
    print_arr(a+1, size-1);
}
```

Write a function that prints out an array of integers of odd size in the following order: a[0], a[n-1], a[1], a[n-2], ..., a[(n-1)/2] (so if a is {1,2,3,4,5}, we print out 1 5 2 4 3) without using loops.

```c
void print_arr2(int a[], int size)
{
    if(size == 1){
        printf("%d", a[0]);
        return;
    }

    printf("%d %d ", a[0], a[size-1]);
    print_arr2(a+1, size-2);
}
```

Write a function exists_sorted_column, with the given header

```c
int exists_sorted_column(int a[N][N])
```

(N is `#defined` somewhere)

, which returns 1 if the array int a[N][N] contains a column which is strictly increasing and 0 otherwise.

Example:

```c
int a[N][N] = {{1, 2, 5},
               {0, 6, 5},
               {2, 10, 7}};
```

the middle column is strictly increasing, so we should return 1. If the 6 in the centre were changed to 1, then we would have to return 0.
[aside: how to figure out what a[1][2] is? A is like an array of arrays, so a[1] is {0, 6, 5} and a[1][2] is now 5.]

Solution:
We need to check all the columns to see whether they are increasing. So we need to loop over the columns. Now to see whether a column is increasing, we need to loop over the rows in a particular column.

```c
int exists_sorted_column(int a[N][N])
{
    int i, j;
    for (j = 0; j < N; j++){
        for (i = 1; i < N; i++)
            if ( a[i][j] <= a[i-1][j] )
                break;
        if (i == N) return 1;
    }
    return 0;
}
```

Given two unsigned ints which don’t contain the digit 0, determine whether the second is a permutation of the first. For example, 312 is a permutation of 123. 3122 is a permutation of 2123, but 122 is not a permutation of 21.

```c
int is_permutation(unsigned int n1, unsigned int n2) {
    int digits[10] = {0};
    int i;

    while (n1 > 0) {
        digits[n1%10]++;
        n1 /= 10;
    }
    while (n2 > 0) {
        digits[n2%10]--;
        n2 /= 10;
    }
    for (i=0; i<10; i++) {
        if (digits[i])
            return 0;
    }
    return 1;
}
```
Write a function that multiplies two polynomials of order n

```c
void mult_poly(double *z, double *x, double *y, int n)
{
    int i, j;
    for (i = 0; i <= 2*n; i++)
        z[i] = 0;
    for (i = 0; i <= n; i++)
        for (j = 0; j <= n; j++)
            z[i+j] += x[i] * y[j];
}
```

*Source acknowledgement: the Technion’s 234114 final exam from Spring 2001A, Winter 2006A; CSC270 midterm, Fall 2002*