Module 4 solutions

1. See quiz answers here.
2. The program computes the average of the 10 numbers in the array, and it can be rewritten as follows:

```c
#include <stdio.h>

int main(void) {
    int arr[10] = {19,74,13,67,44,80,7,36,9,77};
    int i = 0;
    double val = 0.0;
    for (i=0;i<10;i++) val += arr[i];
    val /= 10;
    printf("Value: %.2f\n",val);
    return 0;
}
```

3. The condition number of $f(x) = \cos(x)$ at x is

$$\operatorname{cond}(f,x) = \left|\frac{xf'(x)}{f(x)}\right| = \left|\frac{x\sin(x)}{\cos(x)}\right| = |x\tan(x)|.$$

4. The derivative of f is

$$f'(x) = \begin{cases} \frac{1}{2} & x = 0, \\ \frac{xe^x - e^x + 1}{x^2}, & x \neq 0, \end{cases}$$

and the relative condition number is

$$\operatorname{cond}(f,x) = \frac{|xf'(x)|}{|f(x)|} = \begin{cases} 0, & x = 0, \\ \frac{|xe^x - e^x + 1|}{|e^x - 1|}, & x \neq 0. \end{cases}$$

It follows from the fact that $\operatorname{cond}(f,x)$ is continuous that the problem of evaluating f near 0 is well-conditioned.

The subtraction, $e^x - 1$, will be subject to severe cancellation when x is near 0, but the problem is well-conditioned, and hence the implementation is numerically unstable. Cancellation can be avoided by using the function `expm1()` (defined in `math.h`), which evaluates $\exp(x) - 1$ directly (i.e., without evaluating $\exp(x)$ first).

5. The relative condition number for the sum is approximately $500/0.01 = 5 \cdot 10^4$, and hence sequential summation yields a relative forward error of no more than $5 \cdot 10^4 \cdot 1.06nu = 5.3 \cdot 10^4 nu$.
6. Autolab solutions will be available on DTU Learn after the deadline.
7. Exercise 3 (Ch. 11, p. 255):

```c
void avg_sum(double a[], int n, double *avg, double *sum) {
    *sum = 0.0;
    for (int i=0;i<n;i++) *sum += a[i];
    *avg = *sum/n;
}
```

Exercise 6 (Ch. 11, p. 255-256):

```c
void find_two_largest(int a[], int n,
    int *largest, int *second_largest) {
    if (n < 2) return;
    *largest = a[0] >= a[1] ? a[0] : a[1];
    *second_largest = a[0] >= a[1] ? a[1] : a[0];
    for (int i=2;i<n;i++) {
        if (a[i] > *largest) {
            *second_largest = *largest;
            *largest = a[i];
        }
        else if (a[i] > *second_largest)
            *second_largest = a[i];
    }
}
```

8. Exercise 6 (Ch. 12, pp. 273-274):

```c
int sum_array(const int a[], int n) {
    int sum=0, *p;
    for (p=a; p<a+n; p++)
        sum += *p;
    return sum;
}
```

Exercise 9 (Ch. 12, pp. 274):

```c
double inner_product(const double *a, const double *b, int n) {
    double sum = 0.0, *pa=a, *pb=b;
    while (pa < a+n)
        sum += *(pa++) * *(pb++);
    return sum;
}
```