

SFWR ENG 2S03 — Principles of Programming

26 October 2006

Exercise 6.2 — Please finish!

Exercise 7.1 — Array Mapping (45% of Midterm 4, 2003 — new (d))

Let the following functions be given:

```
double square(double x) { return x * x; }  
double sqrt(double x);          /* from math.h */
```

You are to produce a function *arrayMap* that has the following features:

- It takes as arguments a one-dimensional array *a*, the size *n* of the array *a*, and a function *f* that could be for example *square* or *sqrt* from above.
- For each array element, *arrayMap* applies its argument function *f* to the contents of this array element, and then stores the result returned by *f* back in that array element.
- Each time the call to *f* sets *errno* to a non-zero value, the library function *perror* is used to print an informative message **that includes the current array index**.

```
void perror(const char *s);
```

The routine *perror()* produces a message on the standard error output, describing the last error encountered during a call to a system or library function. First [...] the argument string *s* is printed, followed by a colon and a blank. Then the message and a new-line.

- *arrayMap* returns the number of calls to *f* that set *errno* to a non-zero value.

- (a) ≈5% Produce a prototype for *arrayMap*.
- (b) ≈20% Implement *arrayMap*.
- (c) ≈20% **(independent of (a) and (b)!**

Assume *arrayMap* to be given, and also the following function for some kind of initialisation of an array of doubles:

```
void arrayInit(double ar[], int size);
```

Write a *main* function that contains a 3×10 array of doubles, initialises the first row through a call to the function *arrayInit*, copies the first row to the second and third rows using the library function *memmove*, and finally uses *arrayMap* and the functions from above to square all elements of the second row, and to change all elements of the third row to their respective square roots. After each call to *arrayMap*, the number of errors should be displayed.

```
void * memmove(void * dest, const void * src, size_t n);
```

The *memmove()* function copies *n* bytes from memory area *src* to memory area *dest*.

- (d) **New** Produce a variant *arrayMapNew* that does not change its argument in-place, but produces a new array to store the results of the argument function applications.