Technical University of Denmark

Written examination, December 9, 2015	Page 1 of 10 pages
Course name:	Mathematical software programming
Course number:	02635
Aids allowed:	All aids allowed
Exam duration:	4 hours
Weighting:	80/100

Final exam
Mathematical Software Programming

This exam contains a total of 20 questions: 16 multiple choice questions (questions 1–16) and 4 programming questions (questions 17–20). Your exam answers must be submitted electronically as a **PDF document**. You may include your code in the document along with your answers or submit the code separately in a ZIP file.

1. (2 points) The C language uses what method to pass function arguments?

   A. *Call-by-pointer.*

   B. *Call-by-reference.*

   C. *Call-by-value*, but pointers can be used to simulate *call-by-reference.*

   D. *Call-by-reference*, but pointers can be used to simulate *call-by-value.*

2. (2 points) Which header file should be included to use memory allocation functions such as `malloc` and `calloc`?

   A. `stdlib.h`

   B. `stdio.h`

   C. `memory.h`

   D. `alloc.h`

3. (2 points) What does the program below print?

```c
#include <stdio.h>
void myfunc(int * p){ p++; }
int main(){
    int i[3] = {0,1,2}, *pi = NULL;
    pi = i;
    myfunc(i);
    myfunc(pi);
    printf("%d\n", *pi);
    return 0;
}
```

   A. 0

   B. 1

   C. 2

   D. NULL

4. (4 points) Suppose that `arr` is a variable of type `double *` that points to the first element of a row-major representation of an $m \times n$ matrix (i.e, `arr` points to the first of mn elements which are stored consecutively in memory).

   (a) What is the *stride* of the elements corresponding to a row of the matrix?

   A. 1
   B. m
   C. n
   D. mn

   (b) What is the *stride* of the elements corresponding to a column of the matrix?

   A. 1
   B. m
   C. n
   D. mn

5. (2 points) Which of the following lines of code correctly allocates storage for a `double` array of length n?

   A. `double *p = (double *) malloc(n);`
   B. `double p = (double) malloc(n);`
   C. `double *p = (double *) malloc(n*sizeof(double));`
   D. `double p = (double) malloc(n*sizeof(double));`

6. (2 points) What does the term *memory leak* refer to?

   A. Calling `malloc` twice.
   B. Calling `free` twice.
   C. Failing to release automatically allocated memory.
   D. Failing to release dynamically allocated memory.

7. (2 points) Suppose the variable `p` is a pointer to a structure with members `a` and `b`. Which of the following operators is used to access the two members?

   A. The operator `&` (i.e., `p&a` and `p&b`).
   B. The operator `*` (i.e., `p*a` and `p*b`).
   C. The operator `->` (i.e., `p->a` and `p->b`).
   D. The operator `.` (i.e., `p.a` and `p.b`).

8. (2 points) Consider the following code:

```
double sum=0;
for (int i=0;i<n;i++)
    sum += arr[i];
```

The references to `arr` are ...

A. temporally local

B. spatially local

C. both temporally and spatially local

D. neither temporally nor spatially local

9. (2 points) A cache miss refers to ...

A. a system without cache memory

B. a system with a single level of cache memory

C. a failed attempt to copy data from the main memory into the cache

D. a failed attempt to read or write a piece of data in the cache

10. (2 points) What will happen if you assign a value to an array element whose index exceeds the size of the array?

A. The compiler will issue a warning.

B. The behavior is undefined, and the program may crash.

C. The size of the array grows.

D. The element is set to 0.

11. (2 points) When parallelizing a program, the use of resources (e.g. CPU time) typically ...

A. increases

B. decreases

C. stays the same

12. (2 points) Consider the following piece of code:

```
double a = 0.5, b = a, c = 1.0e-16;
a += c;
a -= c;
b -= c;
b += c;
```

What are the values of `a` and `b`?

A. `a` and `b` are both equal to 0.5.

B. `a` is equal to 0.5 and `b` is less than 0.5.

C. `a` is less than 0.5 and `b` is equal to 0.5.

D. `a` is less than 0.5 and `b` is less than 0.5.

13. (2 points) A class in C++ is ...

A. a definition of an abstract data type

B. an abstract variable

C. an instance of an object

D. a pointer to a data structure

14. (2 points) In object-oriented programming, an *object* refers to ...

A. a class

B. a structure

C. an instance of a class

D. a class with one or more member functions

15. (2 points) Supose that a list of length n is implemented using a dynamic array. What is the complexity of inserting an element at position 0 of the list?

A. $O(1)$

B. $O(\log n)$

C. $O(n)$

D. $O(n^2)$

16. (6 points) A half-precision floating point number occupies 16 bits and has the following representation

s	$e_1 \dots e_5$	$d_1 d_2 \dots d_{10}$

where s is the sign bit, d_i is the ith bit of the mantissa, and e_i is the ith bit of the exponent. Thus, a half-precision floating point number can be represented as

$$x = (-1)^s \cdot (d_0.d_1 d_2 \dots, d_{10})_2 \cdot 2^E = (-1)^s \cdot \sum_{i=0}^{10} d_i 2^{E-i}$$

where $E \in \{-14, -13, \dots, 14, 15\}$ is a decimal representation of the exponent.

(a) The representation of x is called normal if ...

A. the exponent E is equal to zero

B. the exponent E is equal to one

C. the implicit bit d_0 is equal to zero

D. the implicit bit d_0 is equal to one

(b) What is the largest number that can be represented using the half-precision floating point format?

A. 32,752

B. 65,504

C. 65,520

D. 65,535

(c) What is the machine epsilon for the half-precision floating point format?

A. $\epsilon = 2^{-9}$

B. $\epsilon = 2^{-10}$

C. $\epsilon = 2^{-11}$

D. $\epsilon = 2^{-12}$

17. (5 points) Write a function that computes the difference of neighboring elements of a vector $x = (x_1, \ldots, x_n)$ and returns the result $y = (y_1, \ldots, y_{n-1})$ where

$$y_i = x_{i+1} - x_i, \quad i = 1, \ldots, n-1.$$

Your function should have the following prototype:

```c
double * diff(double *x, int n);
```

The first input x should be a pointer to the first element of an array, and the second input n represents the length of the array. The output should be a pointer to the first element of an array that contains the result.

Explain how you tested your function.

18. (9 points) The angle θ between two vectors x and y of length n can be computed from the identity

$$x^T y = \cos(\theta)\|x\|_2\|y\|_2$$

where

$$x^T y = \sum_{i=1}^{n} x_i y_i, \qquad \|x\|_2 = \left(\sum_{i=1}^{n} x_i^2\right)^{1/2}.$$

A programmer wrote the following function to compute the angle (in radians):

```
double angle(double *x, double *y, int n) {

    double norm_x = 0.0, norm_y = 0.0, dot = 0.0;
    int i;

    for (i=1;i<=n;i++) {
        norm_x += x[i]*x[i];
        norm_y += y[i]*y[i];
        dot += x[i]*y[i];
    }
    norm_x = sqrt(norm_x);
    norm_y = sqrt(norm_y);

    return acos(dot/(norm_x*norm_y));
}
```

(a) There are some problems with this function. What are they?

(b) Fix the problems and implement a new `angle` function.

(c) Consider the memory access pattern in the loop inside the `angle` function. What can be said about locality?

19. (12 points) An nth order polynomial

$$p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$$

may be represented using an array of length $n+1$, corresponding to the $n+1$ coefficients $a_n, a_{n-1}, \ldots, a_0$. The derivative of the polynomial, $p'(x)$, is also a polynomial, and its degree is $n-1$.

(a) Define a data structure that represents a polynomial of order n. You may assume that n is an integer that is greater than zero.

You may use the following template:

```
struct polynomial {

};
```

(b) Write a function that takes a polynomial $p(x)$ as input and returns the derivative $p'(x)$ as output. You may use the following function prototye:

```
struct polynomial derivative(struct polynomial poly);
```

(c) How did you test your code to verify its correctness?

20. (16 points) The binomial coefficient $\binom{n}{k}$, or n choose k, is defined as

$$\binom{n}{k} = \begin{cases} \frac{n!}{k!(n-k)!} & n \geq k \geq 0 \\ 0 & k < 0 \text{ or } k > n \end{cases} \tag{1}$$

where n and k are integers. Alternatively, the binomial coefficient can be defined recursively as

$$\binom{n}{k} = \begin{cases} \binom{n-1}{k-1} + \binom{n-1}{k} & n > k > 0 \\ 1 & n > k, k = 0 \quad \text{or} \quad n = k, k \geq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

(a) Implement the binomial function based on (1). Your function should have the following prototype:

```
long binomial_v1(long n, long k);
```

(b) Implement a recursive binomial function based on (2). Your function should have the following prototype:

```
long binomial_v2(long n, long k);
```

(c) How did you test your implementations of the binomial function to ensure its correctness?

(d) Suppose that you want to compute $\binom{2k}{k}$. What is the time-complexity of this computation if you use `binomial_v1`? What is the time-complexity of this computation if you use `binomial_v2`?