02635 — Trial Exam
Matematical Software Programming

This trial exam consists of a total of 18 questions: 14 multiple choice questions (questions 1–14) and 4 programming questions (questions 15–18).

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) Calling `free` twice with the same address is ok.

   A. True

   [✓ B. False]

2. (2 points) A line that begins with a hashtag (`#`) is called a ...

   A. pragma

   B. header guard

   [✓ C. preprocessor directive]

   D. operator

3. (2 points) What is loop-unrolling?

   A. A technique that can be used to avoid loops.

   [✓ B. A basic compiler optimization.]

   C. A recursive method that eliminates loops.

   D. A method for dynamically allocating loop variables.

4. (2 points) Suppose the variable `s` is a structure with members `a` and `b`. Which of the following operators is used to access the two members?

   A. The operator `&` (i.e., `s&a` and `s&b`).

   B. The operator `*` (i.e., `s*a` and `s*b`).

   [✓ C. The operator `->` (i.e., `s->a` and `s->b`).]

   D. The operator `.` (i.e., `s.a` and `s.b`).

5. (2 points) Consider the following code:

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

The references to `sum` are ...

[✓ A. temporally local]

B. spatially local

C. both temporally and spatially local

D. neither temporally nor spatially local

6. (2 points) Catastrophic cancellation may occur when ...

[✓ A. subtracting two nearly equal numbers]

B. subtracting a number close to zero from a large number

C. multiplying to nearly equal numbers

D. multiplying a large number by a number close to zero

7. (2 points) Consider the following piece of code:

```c
double a = 1.0, 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 1.0.

B. `a` is equal to 1.0 and `b` is less than 1.0.

[✓ C. `a` is less than 1.0 and `b` is equal to 1.0.]

D. `a` is less than 1.0 and `b` is less than 1.0.

8. (2 points) The speed-up when parallelizing a task can be expressed as

$$S(p) = \frac{\text{wall-time on 1 CPU core}}{\text{wall-time on } p \text{ CPU cores}}$$

How is Amdahl's law related to the speed-up?

A. Amdahl's law provides the expected speed-up

B. Amdahl's law provides the worst-case speed-up

C. Amdahl's law provides a theoretical lower-bound of the speed-up

D. Amdahl's law provides a theoretical upper-bound of the speed-up

9. (6 points) Consider the following piece of C++ code:

```cpp
std::vector<double> v;
std::cout << v.size() << std::endl;
v.push_back(1.0);
```

(a) What is the name of the `::` operator?

[✓ A. The *scope resolution* operator.]

B. The *member access* operator.

C. The *address of* operator.

D. The *dereferencing* operator.

(b) The variable `v` is ...

A. an array of type `double`

B. a pointer to an array of type `double`

C. a `std::vector<double>` class

[✓ D. an instance of the `std::vector<double>` class]

(c) What will be printed on the screen?

A. Nothing will be printed.

[✓ B. 0]

C. 1

D. 2

10. (2 points) Operator overloading in C++ allows the programmer to ...

    A. redefine the meaning of built-in operators using macros

    B. define the meaning of built-in operators when applied to abstract data types

    C. change how operators work on built-in data types

    D. define the behavior when an object is constructed

11. (2 points) What is the time complexity of adding two arrays of length n?

    A. $O(1)$

    B. $O(\log n)$

    C. $O(n)$

    D. $O(n^2)$

12. (2 points) A list is an abstract data type. What characterizes a list?

    A. Elements can be accessed, inserted, and deleted only at the ends of the list.

    B. Elements can be accessed, inserted, and deleted only at one end of the list.

    C. Elements can be accessed, inserted, and deleted at any position.

13. (2 points) Unlike a structure in C, a structure in C++ may ...

    A. contain member functions and access specifications

    B. contain member functions

    C. contain access specifications

    D. contain a pointer to a function

14. (2 points) What is the time complexity of copying an array of length n?

    A. $O(1)$

    B. $O(\log n)$

    C. $O(n)$

    D. $O(n^2)$

15. (12 points) Numerical differentiation can be used to approximate the derivative of a differentiable function $f(x)$. Recall that the derivative of f at x is defined as

$$f'(x) = \frac{df(x)}{dx} = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h}$$

which suggests the following approximation scheme:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}, \quad h > 0.$$

This is known as a forward-difference approximation of the derivative of f at x.

(a) Write a function that approximates the derivative of

$$f(x) = \frac{2x^2 - 4x}{2x^2 + 2x + 3}$$

using the forward-difference approximation. Use the following prototype:

```
double df(double x, double h);
```

(b) How did you test your implementation to ensure its correctness?

(c) Write a short program that uses your function `df` to approximate the derivative of f at $x = 0.6$. Report your result with four significant digits and explain how you choose the parameter h.

16. (12 points) A programmer wrote the following code to convert Cartesian coordinates (x, y) to polar coordinates (r, θ):

```
struct polar_coordinate {
   double theta;
   double r;
};

struct polar_coordinate * polar(double x, double y) {
    struct polar_coordinate pc;
    pc.r = sqrt(x*x + y*y);      // compute radius
    pc.theta = atan2(y, x);      // compute angle
    return &pc;
}
```

(a) There is a bug in this function. What is the problem?

(b) Rewrite the `polar` function to fix the problem.

(c) How did you test your implementation to ensure correctness?

17. (12 points) An $n \times n$ tridiagonal matrix of the form

$$A = \begin{pmatrix} \alpha & \beta & & & \\ \beta & \alpha & \beta & & \\ & \ddots & \ddots & \ddots & \\ & & \beta & \alpha & \beta \\ & & & \beta & \alpha \end{pmatrix}, \tag{1}$$

where α and β are nonzero constants, arise in a number of engineering applications. The matrix–vector product $y = Ax$ can easily be computed without explicitly storing A, i.e., if x and y are vectors of length n,

$$y_i = \begin{cases} \alpha x_i + \beta x_{i+1} & i = 1 \\ \beta x_{i-1} + \alpha x_i + \beta x_{i+1} & i = 2, \dots, n-1 \\ \beta x_{i-1} + \alpha x_i & i = n. \end{cases}$$

(a) Write a function `tdmv` that computes the matrix–vector product Ax where A is a matrix of the form (1) of order n and x is a vector of length n. Use the following prototype:

```
double * tdmv(int n, double alpha, double beta, double * x);
```

The input `x` should be a pointer to the first element of an array of length n, and the values `alpha` and `beta` define the matrix A. The return value should be a pointer to the first element of an array of length n that contains the result.

(b) How did you test your implementation to verify its correctness?

(c) The matrix A can be used to approximate the second derivative of a continuous function $f(x)$, i.e., using the approximation

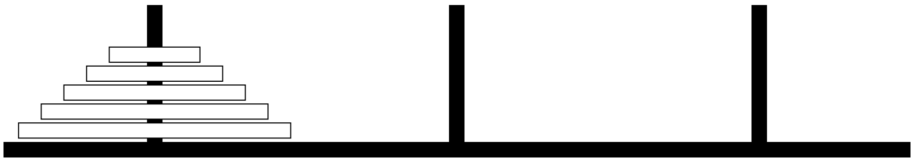
$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

we can express the approximation at $N+1$ points $(\bar{x}, \bar{x}+h, \dots, \bar{x}+Nh)$ as

$$\begin{pmatrix} f''(\bar{x}) \\ f''(\bar{x}+h) \\ \vdots \\ f''(\bar{x}+(N-1)h) \\ f''(\bar{x}+Nh) \end{pmatrix} \approx \frac{1}{h^2} \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix} \begin{pmatrix} f(\bar{x}) \\ f(\bar{x}+h) \\ f(\bar{x}+2h) \\ \vdots \\ f(\bar{x}+(N-2)h) \\ f(\bar{x}+(N-1)h) \\ f(\bar{x}+Nh) \end{pmatrix}.$$

Write a short program that uses the function `tdmv` to compute an approximation to the second derivative of $f(x) = x\cos(x)$ for $x \in [0, \pi]$ using 50 points.

18. (12 points) The *Tower of Hanoi* is an old mathematical puzzle. You are given three pegs and n discs of increasing size, and the discs are initially ordered by size and placed on one of the pegs as illustrated in the figure below.



The goal is to move all discs to another peg (say, from peg 1 to peg 3), obeying the following rules: only one disc can be moved at a time and a disc may never be placed on top of a smaller disc.

Suppose we label the disc 1 through n where 1 corresponds to the smallest disc and n corresponds to the largest disc. The puzzle can now be solved recursively using the following function:

```c
void hanoi(int n, int from_peg, int to_peg) {
   int other_peg;

   assert(n > 0);
   assert(from_peg < 4 && from_peg > 0);
   assert(to_peg < 4 && to_peg > 0);

   if (!(from_peg == 1 || to_peg == 1))
      other_peg = 1;
   else if (!(from_peg == 2 || to_peg == 2))
      other_peg = 2;
   else
      other_peg = 3;

   if (n == 1)
      printf("Move disc %d from peg %d to peg %d\n",
               1, from_peg, to_peg);
   else {
      hanoi(n-1, from_peg, other_peg);
      printf("Move disc %d from peg %d to peg %d\n",
               n, from_peg, to_peg);
      hanoi(n-1, other_peg, to_peg);
   }
   return;
}
```

(a) Analyze the `hanoi` function. How many moves (i.e., function calls) are necessary to solve to puzzle with n discs?

(b) What is the space complexity if you use the `hanoi` function to solve the problem?