Chair of Scientific Computing in Computer Science
School of Computation, Information and Technology
Technical University of Munich

Eexam
Place student sticker here

Note:

- During the attendance check a sticker containing a unique code will be put on this exam.
- This code contains a unique number that associates this exam with your registration number.
- This number is printed both next to the code and to the signature field in the attendance check list.

Advanced Programming

Exam:	IN1503 / Endterm	**Date:**	Monday 13th February, 2023
Examiner:	Prof. Hans-Joachim Bungartz	**Time:**	13:00 – 14:15

	P 1	P 2	P 3	P 4
I				

Working instructions

- Even though problems 2-4 follow the same theme, they are independent and can be solved in any order. Often, subproblems are also independently solvable, so make sure to try everything.
- You do not need to specify required header inclusions or the main function, unless explicitly asked.
- Allowed resources:
  - one **hand-written sheet of A4 paper (both sides)** with notes
  - one **analog dictionary** English ↔ native language
- Answers are only accepted if the solution approach is documented. Give a reason for each answer unless explicitly stated otherwise in the respective subproblem.
- This exam consists of **12 pages** with a total of **4 problems**.
  Please make sure now that you received a complete copy of the exam.
- The total amount of achievable credits in this exam is 42 credits.
- Detaching pages from the exam is prohibited.
- Do not write with red or green colors nor use pencils.
- Physically turn off all electronic devices, put them into your bag and close the bag.

Left room from ________ to ________ / Early submission at ________

Problem 1 Warming up (8 credits)

In all questions, select only one answer. If you want to edit your answer, fill the box of your previous answer completely. This problem is automatically graded, but you can check the grading during the review period.

a) In the expression `int a = 1; auto b = a++ / ++a;`, what is the type and value of `b`?

- ☐ `int` and 0
- ☐ `double` and 0.66
- ☐ `char` and 1
- ☐ `float` and 0.5f

b) Which one of the following commands will compile and execute a C++ code `main.cpp` with GCC on Linux?

- ☐ `g++ main.cpp -o main.exe && ./main.exe`
- ☐ `gcc main -e main.exe && ./main.exe`
- ☐ `gcc main.cpp && ./main`
- ☐ `g++ -c main.cpp && ./main.o`

c) Objects of type `std::unique_ptr` are stored:

- ☐ On the heap
- ☐ On the free store
- ☐ On the reference
- ☐ On the stack

d) Which of the following can we **not** pass to a `&&` argument of a templated function `void f(T &&)`?

- ☐ the result of `std::make_unique<int>(a)`, where `int a = 0`
- ☐ the literal `3.14`
- ☐ the result of `std::move(b)`, where `double b = 1.23`
- ☐ the variable `x`, where `int x = 2`.

e) Given an abstract base class `Fruit` and the derived classes `Apple` and `Pear`, which of the following could let the user decide between types of fruits at runtime?

- ☐ `std::unique_ptr<Fruit> fruit = std::make_unique<Fruit>();`
- ☐ `std::unique_ptr<Apple> fruit = std::make_unique<Apple>();`
- ☐ `auto fruit = std::make_unique<Fruit>();`
- ☐ `std::shared_ptr<Fruit> fruit = std::make_shared<Apple>();`

f) Consider the code:

```
#include <iostream>

int f(int c){
    return c + 1;
}

int main() {
    int a{42};
    const auto b = &a;
    f(*b); // What is returned here?
    a;     // What is the value of a here?
}
```

Does the code compile?
If yes, what is the returned value of `f(*b)` and the value of `a` after that call?

- ☐ The code does not compile
- ☐ The code compiles and results to `f(*b) = 43` and `a = 42`
- ☐ The code compiles and results to `f(*b) = 42` and `a = 42`
- ☐ The code compiles and results to `f(*b) = 43` and `a = 43`

g) Consider a class `A` with all basic operators defined and no members. In `main()`, we define:

```
A a1;
A a2 = a1;
```

Which operator is called at line 2? Assume no optimizations.

- ☐ Move constructor
- ☐ Copy assignment operator
- ☐ Copy constructor
- ☐ Move assignment operator

h) Consider the following code:

```
template<typename T> concept Eddible = requires(T x) { x.eat(); };

template<Eddible T> void eat(T food){ food.eat(); }

struct Berry     { void eat(bool scary = false);              };
struct Chocolate { void eat();                                };
struct Pancake   { void eat(Chocolate& topping);              };
struct Soup      { std::string eat(){return "mmm... soup!";}; };
```

For which of the following statements will the compiler complain that the object is not `Eddible`?

- ☐ `Pancake pancakeChoco; eat(pancakeChoco);`
- ☐ `Berry pumkin; eat(pumkin);`
- ☐ `Chocolate choco; eat(choco);`
- ☐ `const Soup noodleSoup; eat(noodleSoup);`

Problem 2 Data types, functions, STL (11 credits)

Multiple energy providers are selling electricity contracts in Advprogland. Each provider offers a different price per kWh. During the energy crisis, the government of Advprogland decides that there should be an upper limit on the price.

0
1
2

a) Write a complete C++ program which:
- Stores a variable `priceCap`, set to 0.20EUR/kWh
- Displays the value `priceCap` to the user using streams, in a message ended by a new line
- Returns a successful status code
- Includes any necessary header files

0
1

b) For each provider, we know its name and the price associated with it. We don't know how many providers we will have in total. Declare a single, contiguous, dynamically-allocated container of associated name-price pairs to later store information about energy providers. Do not define any class or struct.

0
1

c) Add three providers to the container declared in task b):
- Name: A, price: 0.15
- Name: B, price: 0.42
- Name: C, price: 0.14

d) Using a ranged-for loop, reduce the price of all providers (as declared in task b)) to the maximum allowed price (`priceCap`).

0
1
2

e) Consider the following form of the `std::sort` algorithm:

```
template< class RandomIt, class Compare >
constexpr void sort( RandomIt first, RandomIt last, Compare comp );

// comp is a comparison function object which returns true if the first argument
// is less than (i.e. is ordered before) the second.
// The signature of the comparison function should be equivalent to the following:
bool cmp(const Type1 &a, const Type2 &b);
```

Using `std::sort`, sort the container of providers (declared in task b)) from the cheapest to the most expensive. You do not need to print the elements.

0
1
2
3

f) Print the name of the cheapest provider in the container (consider already sorted by task e)). Access the container with a method that will throw an exception in case there are no elements in the container.

0
1

Problem 3 Object-oriented programming (13 credits)

In Advprogland, multiple power plants of different types are installed. Each power plant has limitations in the power it can produce, while the power of some (such as wind turbines) depends on the current conditions.

0 1 2

a) Define a class `PowerPlant` with:

- a member variable `_maxPower` of type **`double`** to store its design power. The member variable should never be changed and should only be accessible by this class and by potential subclasses;
- a constructor receiving a parameter to set the member variable (only declare, do not define).

0 1

b) The class `PowerPlant` from task a) shall be extended. Indicate the declaration of an abstract virtual method `computeExpectedPower()` without any parameters that shall never modify any member and shall return a double value.

0 1

c) Can we instantiate objects of the class type `PowerPlant` from tasks a) and b)? Explain.

d) Implement an additional class `WindTurbine` which inherits from `PowerPlant` from task a). The new class `WindTurbine` should:

- have two additional private, never-changing member variables of type **`double`** to store the density of air (to a value of 1.225 kg/m^3) and the diameter of the rotor.
- declare a constructor that initializes the complete state of the object via parameters (will implement in task e));
- declare the function `computeExpectedPower()`, ensuring that it fits the specification of the base class (will implement in task f))

0 1 2

e) Provide the constructor implementation for the class `WindTurbine` from task d), **as if it was defined in a separate .cpp file** .

0 1 2

0
1

f) Implement the method `computeExpectedPower()` of the class `WindTurbine` such that it returns 60 % of the maximum power of a wind turbine as a rough estimation for an average expected power output.

0
1
2

g) Implement a method `computeCurrentPower()` for the class `WindTurbine` which
- returns a **`double`** value of the current power value depending on the wind speed v as an input parameter of type **`double`**;
- computes the power $p(v)$ in Watt according to the following formula:

$$p(v) = \frac{1}{2}\rho \cdot A \cdot v^3$$

where ρ is the density of the air and A represents the cross-sectional area of the rotor ($A = \pi \cdot r^2$).
Make sure that the output of `computeCurrentPower()` never exceeds the maximum power of the wind turbine!

0
1

h) Which of the two methods `computeExpectedPower()` and `computeCurrentPower()` could be marked as **`constexpr`**? Why?

Problem 4 Performance (10 credits)

As part of your job, you are requested to calculate the total amount of money to be paid for the power consumption of NB buildings. Each building has a different energy provider, and thus the cost per kWh differs.

a) For this task, you are given the total power consumption of each building for the last NM months. After some thought, you come up with this code:

0 1 2 3

```cpp
std::array<float, NB> price_list;
std::array<std::array<float, NM>, NB> consumptions;

// Initialization of price_list, consumptions ...

float total_cost = 0;
for (int i = 0; i < consumptions.size(); i++) {        // Buildings
  for (int j = 0; j < consumptions[i].size(); j++) {   // Months
    total_cost += consumptions[i][j] * price_list[i];
  }
}
```

Give the computational intensity of the code in terms of NM and NB, with **units**.

0
1

b) The code from task a) will run on a machine with the following characteristics:

- Peak performance: 2 GFLOP/s
- Memory bandwidth (b_s): 1 GByte/s

For a particular value of `NM` and `NB`, you find that the computational intensity is 0.5 FLOP/Byte. What floating point performance will you expect from your code? Justify your answer.

0
1

c) Consider again the code from task a). If we assume that one cache line can hold `NUM_MONTHS` elements, is the code using the cache efficiently? Justify your answer.

0
1

d) Due to an unexpected energy crisis, the government has decided to subsidize the energy price for the first `K` months (`K < NM`). During this period, consumers will only have to pay for 80% of the total consumed power.

One of your colleagues proposes a completely new code to calculate the total bill:

```
std::list<float> price_list;
std::array<std::array<float, NM>, NB> consumptions;

// Initialization of price_list, consumptions ...

float cost = 0;
for (int i = 0; i < consumptions.size(); i++) {
  for (int j = 0; j < consumptions[i].size(); j++) {
    // function get returns the value of the i-th element of the list
    if (j < K)
      cost += consumptions[i][j] * get(price_list, i) * (80.0 / 100.0);
    else
      cost += consumptions[i][j] * get(price_list, i);
  }
}
```

After compiling the code with the `-O3` flag, would you expect this code to be vectorized? Why?

e) Consider running the code from task d) with a processor that does not offer vector instructions. Indicate **two** changes to optimize its performance. Give a **code snippet** for each change, to show how the code should be altered. You can refer to specific lines to replace.

☐ 0
☐ 1
☐ 2
☐ 3

f) A colleague of yours suggests to change `80.0 / 100.0` to `0.8` to avoid one division. Will this affect the performance if both versions are compiled with `-O3`? Justify your answer.

☐ 0
☐ 1

Additional space for solutions–clearly mark the (sub)problem your answers are related to and strike out invalid solutions.