Universität Siegen

In-Class Exercise 1

No pencils, no devices allowed

Advanced Programming in C++ Exercises

Ubiquitous Computing

Name: _____

Student ID: _____

Complete the program below so that it prints out a big triangle with the character 'X', depending on the variable **auto size** (with **size** = 3, 4, ..., 200):

size = 3:
```
  X
 XXX
XXXXX
```

size = 4:
```
   X
  XXX
 XXXXX
XXXXXXX
```

size = 5:
```
    X
   XXX
  XXXXX
 XXXXXXX
XXXXXXXXX
```

etc.

Hint: Use **`std::cout and std::endl;`** to write something to the console

```
#include <iostream>
int main() {
  auto size = 0;
  std::cin >> size;  // the user will always enter a value between 3 and 200 here
```

```
   return 0;
}
```

Universität Siegen

In-Class Exercise 1

No pencils, no devices allowed

Advanced Programming in C++ Exercises

Ubiquitous Computing

Name: _____

Student ID: _____

Complete the program below so that it prints out a big triangle that is tilted left with the character 'X', depending on the variable **auto size** (with **size** = 3, 4, ..., 200):

size = 3:
```
X
XX
XXX
XX
X
```

size = 4:
```
X
XX
XXX
XXXX
XXX
XX
X
```

size = 5:
```
X
XX
XXX
XXXX
XXXXX
XXXX
XXX
XX
X
```

etc.

Hint: Use **`std::cout and std::endl;`** to write something to the console

```
#include <iostream>
int main() {
  auto size = 0;
  std::cin >> size;  // the user will always enter a value between 3 and 200 here









   return 0;
}
```

Universität Siegen

In-Class Exercise 1

No pencils, no devices allowed

Advanced Programming in C++ Exercises

Ubiquitous Computing

Name: _____

Student ID: _____

Complete the program below so that it prints out a big triangle that is tilted right with the character 'X', depending on the variable **auto size** (with **size** = 3, 4, ..., 200):

size = 3:

```
  X
 XX
XXX
 XX
  X
```

size = 4:

```
   X
  XX
 XXX
XXXX
 XXX
  XX
   X
```

size = 5:

```
    X
   XX
  XXX
 XXXX
XXXXX
 XXXX
  XXX
   XX
    X
```

etc.

Hint: Use **`std::cout and std::endl;`** to write something to the console

```
#include <iostream>
int main() {
  auto size = 0;
  std::cin >> size;  // the user will always enter a value between 3 and 200 here









   return 0;
}
```

Universität Siegen

In-Class Exercise 1

No pencils, no devices allowed

Advanced Programming in C++ Exercises

Ubiquitous Computing

Name: _____

Student ID: _____

Complete the program below so that it prints out a big upside-down triangle with the character 'X', depending on the variable **auto size** (with **size** = 3, 4, ..., 200):

size = 3:
```
XXXXX
 XXX
  X
```

size = 4:
```
XXXXXXX
 XXXXX
  XXX
   X
```

size = 5:
```
XXXXXXXXX
 XXXXXXX
  XXXXX
   XXX
    X
```

etc.

Hint: Use **`std::cout and std::endl;`** to write something to the console

```
#include <iostream>
int main() {
  auto size = 0;
  std::cin >> size;  // the user will always enter a value between 3 and 200 here
```

```
   return 0;
}
```

Universität Siegen

In-Class Exercise 2

No pencils, no devices allowed

Advanced Programming in C++ Exercises

Ubiquitous Computing

Name: _____

Student ID: _____

A possible variant of the Fibonacci numbers, **vfib**, could be a sequence of numbers, where the **n**th number is recursively defined as: 1 for **n**=0, 1 for **n**=1, and (1 / vfib(**n-1**))+vfib(**n-2**) for **n** ≥ 2

In the code below, enter in field A the necessary code to create the *recursive* function **vfib** which takes a whole number as its sole parameter, to return the **n**th **vfib** number. In field B, create a two-dimensional array of doubles, with dimensions 2 and 120, for which you can choose the name. In field C, use calls to function **vfib** to fill the array with the first 240 of these **vfib** numbers (the order is unimportant). Note: We do not ask to write something to the console.

```
// Field A: declare and implement the function vfib:









int main() {
  // Field B: declare here a two-dimensional array:


  // Field C: use here the function vfib to store the first 240
  // vfib numbers as explained in the question above, in the array:









   return 0;
}
```

Universität Siegen

In-Class Exercise 2

No pencils, no devices allowed

Advanced Programming in C++ Exercises

Ubiquitous Computing

Name: _____

Student ID: _____

A modification of the Fibonacci numbers, **mfib**, is a sequence of numbers, where the **n**th number is recursively defined as: 2 for **n**=0, 3 for **n**=1, and (1 / mfib(**n-1**)) + (1 / mfib(**n-2**)) for **n** ≥ 2

In the code below, enter in field `I` the necessary code to create the *recursive* function **mfib** which takes a whole number as its sole parameter, to return the **n**th **mfib** number. In field `II`, create a two-dimensional array of doubles, with dimensions 2 and 90, for which you can choose the name. In field `III`, use calls to function **mfib** to fill the array with the first 180 of these **mfib** numbers (the order is unimportant). Note: We do not ask to write something to the console.

```
// Field I: declare and implement the function mfib:









int main() {
  // Field II: declare here a two-dimensional array:


  // Field III: use here the function mfib to store the first 180
  // mfib numbers as explained in the question above, in the array:









   return 0;
}
```

Universität Siegen

In-Class Exercise 2

No pencils, no devices allowed

Advanced Programming in C++ Exercises

Ubiquitous Computing

Name: _____

Student ID: _____

A change of the Fibonacci numbers, **chfib**, could be a sequence of numbers, where the **n**th number is recursively defined as: 4 for **n**=0, 5 for **n**=1, and chfib(**n-1**) + (1 / chfib(**n-2**)) for **n** ≥ 2

In the code below, enter in field 1 the necessary code to create the *recursive* function **chfib** which takes a whole number as its sole parameter, to return the **n**th **chfib** number. In field 2, create a two-dimensional array of floats, with dimensions 2 and 11, for which you can choose the name. In field 3, use calls to function **chfib** to fill the array with the first 22 of these **chfib** numbers (the order is unimportant). Note: We do not ask to write something to the console.

```
// Field 1: declare and implement the function chfib:









int main() {
  // Field 2: declare here a two-dimensional array:


  // Field 3: use here the function chfib to store the first 22
  // chfib numbers as explained in the question above, in the array:









   return 0;
}
```

Universität Siegen

In-Class Exercise 3

No pencils, no devices allowed

Advanced Programming in C++ Exercises

Ubiquitous Computing

Name: _____

Student ID: _____

Create a C++ class named **`Student`** with: 1) a private attribute **`mark`** of type double pointer, 2) a constructor, 3) a destructor, and 4) a public method named **`setMark(double mark)`** which does not return anything. The constructor should have no parameters, and use the **`mark`** attribute to create an array of size 5 (without initializing it). The destructor should print out all contents of **`mark`** , each on a new line, and then free up the memory taken by **`mark`**. The **`setMark(double mark)`** method should fill in its parameter's value in all of **`mark`** 's elements. In the main program, create a **`Student`** object **`s`** and, solely by calling its methods, make the program output five ones.

```
#include <iostream>  // allows std::cout for output

// Declare the class Student and all its methods, without implementing them:









// Implement the methods of class Student:









int main() {



  return 0;
}
```

Universität Siegen

In-Class Exercise 3

No pencils, no devices allowed

Advanced Programming in C++ Exercises

Ubiquitous Computing

Name: _____

Student ID: _____

Create a C++ class named **Employee** with: 1) a private attribute **salary** of type uint16_t pointer, 2) a constructor, 3) a destructor, and 4) a public method named **setSalary(uint16_t salary)** which does not return anything. The constructor should have no parameters, and use the **salary** attribute to create an array of size 12 (without initializing it). The destructor should print out all contents of **salary** , each on a new line, and then free up the memory taken by **salary**. The **setSalary** method should fill in its parameter's value in all of **salary** 's elements. In the main program, create a **Employee** object **e** and, solely by calling its methods, make the program output 12 times 5000.

```
#include <iostream>  // allows std::cout for output

// Declare the class Employee and all its methods, without implementing them:









// Implement the methods of class Employee:









int main() {



  return 0;
}
```

Universität Siegen

In-Class Exercise 4

No pencils, no devices allowed

Advanced Programming in C++ Exercises

Ubiquitous Computing

Name: _____

Student ID: _____

Create a C++ subclass named **`Circle`** of the class **`Shape`** below. It should publicly inherit everything from Shape, and additionally contain a private attribute **`radius`** which is of type double. The sole constructor of **`Circle`** should take three double parameters for initializing the circle's x position, the y position, and the radius, and should do this without statements in the constructor's body.
The class **`Circle`** should also have a friend method **`print`** which does not belong to the class, takes the reference to a **`Circle`** object as a parameter, and prints out the passed object's radius to the console.
Finally, create a Circle object **`c`** in the main function and initialize it with 5 and 7 as the x and y position, and 12 as the radius. Then, pass this object to the **`print`** function.

```
#include <iostream>  // allows std::cout for output
class Shape {  // class representing a shape
 private:
  double xPos, yPos;  // two-dimensional position of the shape
 public:
  Shape(double x, double y) { this->xPos = x; this->yPos = y; }
};

// Declare the class Circle, as described above:









// Implement the method print:









int main() {




  return 0;
}
```

Universität Siegen

In-Class Exercise 4

No pencils, no devices allowed

Advanced Programming in C++ Exercises

Ubiquitous Computing

Name: _____

Student ID: _____

Create a C++ subclass named **Square** of the class **Shape** below. It should publicly inherit everything from Shape, and additionally contain a private attribute **width** which is of type double. The sole constructor of **Square** should take three double parameters for initializing the square's x position, the y position, and the width, and should do this without statements in the constructor's body.
The class **Square** should also have a friend method **print** which does not belong to the class, takes the reference to a **Square** object as a parameter, and prints out the passed object's width to the console.
Finally, create a Square object s in the main function and initialize it with 4 and 8 as the x and y position, and 17 as the width. Then, pass this object to the **print** function.

```
#include <iostream>  // allows std::cout for output
class Shape {  // class representing a shape in a two-dimensional space
 private:
  double x, y;  // two-dimensional position
 public:
  Shape(double xPos, double yPos) { this->x = xPos; this->y = yPos;}
};

// Declare the class Square, as described above:









// Implement the method print:









int main() {




  return 0;
}
```

Universität Siegen

In-Class Exercise 5

No pencils, no devices allowed

Advanced Programming in C++ Exercises

Ubiquitous Computing

Name: _____

Student ID: _____

Complete a C++ function named **checkIPLength** below. This function should throw a **NetworkException** with the message **"Long IP: "** and the **ip** parameter, if the length of **ip** is higher than 15. This function should also throw **0** if the length of the **ip** parameter is zero.
Then, complete the main function so that, in a **for** loop, all elements of the **ip** array there are given to the **checkIPLength** function and exceptions are caught and handled to the **std:cerr** stream so that the output of a **NetworkException** is delivered with **"Net: "** as a prefix, and all other exceptions are caught and delivered to **std::cerr** with **"Error."**.

```
#include <iostream>  // allows std::cerr for output
class NetworkException : public std::exception {
 public:
  NetworkException(const char * msg, std::string ip) { message = msg + ip; }
  const char * what() { return message.c_str(); }
 private:
  std::string message;
};

// Implement the function checkIPLength, as described above:
void checkIPLength(std::string ip) noexcept(false) {



}




int main() {
  std::string ip[3] = { "192.168.0.1", "155.230.255.1024", "" };
  /* this should for instance generate:
    Net: Long IP: 155.230.255.1024
    Error. */









  return 0;
}
```

Universität Siegen

In-Class Exercise 5

No pencils, no devices allowed

Advanced Programming in C++ Exercises

Ubiquitous Computing

Name: _____

Student ID: _____

Complete a C++ function named **checkFileSize** below. This function should throw a **FileSizeException** with the message **"Oversized file: "** and the **filesize** parameter, if **filesize** is higher than 1024. This function should also throw **-1** if the length of the **filesize** parameter is zero or negative.
Then, complete the main function so that, in a **for** loop, all elements of the **fileSizes** array there are given to the **checkFileSize** function and exceptions are caught and handled to the **std:cerr** stream so that the output of a **FileSizeException** is delivered with **"Size: "** as a prefix, and all other exceptions are caught and delivered to **std::cerr** with **"Fault."**.

```
#include <iostream>  // allows std::cerr for output
class FileSizeException : public std::exception {
 public:
  FileSizeException(const char * msg, double filesize) {
    report = msg + std::to_string(filesize);
  }
  const char * what() { return report.c_str(); }
 private:
  std::string report;
};

// Implement the function checkFileSize, as described above:
void checkFileSize(double filesize) noexcept(false) {


}




int main() {
  double fileSizes[3] = { 168.0, 251024.0, 2.0-9.0 };
  /* this should for instance generate:
    Size: Oversized file: 251024.000000
    Fault. */









  return 0;
}
```