Universität Siegen

In-Class Exercise 1

No pencils, no devices allowed

Advanced Programming in C++ Exercises

Ubiquitous Computing

Student ID: |__|__|__|__|__|__|__|__|

Your *full* name: _____

Write an inline function **`underscore`** below that replaces all the **`spaces`** with underscores (_), in a C string (array of characters) that is provided as a sole function parameter. Then complete **`main()`** below, without using *any* **`if`** statement, so that it prints out a shark's fin with the character '**|**', using the variable **width**:

width = 3:

```
  |
  ||
  | |
```

width = 4:

```
  |
  ||
  | |
  |  |
```

width = 5:

```
  |
  ||
  | |
  |  |
  |   |
```

etc.

`main`'s last output should print "**`A_left_leaning_fin`**". *Use **`auto`** for all your variables whenever possible.*

```
#include <iostream>
// declare and implement the underscore function below:
```

```
int main() {
  char title[] = "A left leaning fin"; auto width = 0;
  std::cout << "Enter shark fin's width: ";
  std::cin >> width;  // users will always enter a number larger than 2
```

```
  underscore(title);
  std::cout << title << '\n';
}
```

Universität Siegen

In-Class Exercise 1

No pencils, no devices allowed

Advanced Programming in C++ Exercises

Ubiquitous Computing

Student ID: |__|__|__|__|__|__|__|__|

Your *full* name: _____

Write below an inline function **`capitalT`** that replaces all the small **`t`**s with capital **`T`**s, in a C string (array of characters); The string is the sole function parameter. Then complete the main function below, without using *any* **`if`** statement, so that it prints out an upper-right triangle with the character 'X', using the variable **size**:

size = 2:	size = 3:	size = 4:	size = 5:	etc.
XX	XXX	XXXX	XXXXX	
X	XX	XXX	XXXX	
	X	XX	XXX	
		X	XX	
			X	

main's last output should print "**`This is a Triangle`**". *Use **auto** for all your variables whenever possible.*

```
#include <iostream>
// declare and implement the capitalT function below:









int main() {
  char caption[] = "this is a triangle"; auto size = 0;
  std::cout << "Enter the triangle size: ";
  std::cin >> size;  // users will always enter a number larger than 1









  capitalT(  caption  );
  std::cout << caption << '\n';
}
```

Universität Siegen

In-Class Exercise 1

No pencils, no devices allowed

Advanced Programming in C++ Exercises

Ubiquitous Computing

Student ID: |__|__|__|__|__|__|__|__|

Your *full* name: _____

Write below the inline function **bigI** that replaces all the small **i**s with capital **I**s in a C string (array of characters), which is provided as a sole function parameter. Then complete the main function below, without using *any* **if** statement, so that it prints out a sail with the character '**/**', using the variable **height**:

height = 3:
```
  /
 //
///
```
height = 4:
```
   /
  //
 ///
////
```
height = 5:
```
    /
   //
  ///
 ////
/////
```
etc.

main should print as its last output "**I see a saIl**". *Use **auto** for all your variables whenever possible.*

```
#include <iostream>
// declare and implement the bigI function below:
```

```
int main() {
  char label[] = "i see a sail"; auto height = 0;
  std::cout << "Enter the sail's height: ";
  std::cin >> height;  // users will always enter a number larger than 2
```

```
  bigI(label); std::cout << label << '\n';
}
```

Universität Siegen

In-Class Exercise 2

No pencils, no devices allowed

Advanced Programming in C++ Exercises

Ubiquitous Computing

Student ID: |__|__|__|__|__|__|__|

Your *full* name: _____

Complete below the class declaration, all implementations it promises, and the main function under the respective comments. Make sure that you provide code that will compile in a fully functioning program, and that you adhere to the given instructions (as comments). *Note that the code continues on the reverse side.*

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

// Complete the class declaration with a private array 'marks' of 11 double elem-
// ents and a public static integer 'len', only 1 default constructor and destruct-
// or that are implemented after declaration (no definitions inside the class):

class Student {
 public:
  operator std::string();
```

```
// In the constructor body, fill the array 'marks' with all possible grades (1.0,
// 2.0,3.0,4.0,5.0,1.3,2.3,3.3,1.7,2.7,3.7) with a single for loop and a single if
// condition. Initialize 'len' to -1 in the body:
```

```
// In the conversion operator implentation, return the contents of 'marks' as a
// string. Use std::to_string(). Copy its length in 'len' using length():

// In the destructor, print out the value of 'len', followed by a new line:

// In the main function, use a decomposition declaration and the above conversion
// operator to print out the elements of myStudents, each on a single line:
int main() {
  auto myStudents = std::make_tuple( Student(), Student() );
```

Universität Siegen

In-Class Exercise 2

No pencils, no devices allowed

Advanced Programming in C++ Exercises

Ubiquitous Computing

Student ID: |__|__|__|__|__|__|__|

Your *full* name: _____

Complete below the class declaration, all implementations it promises, and the main function under the respective comments. Make sure that you provide code that will compile in a fully functioning program, and that you adhere to the given instructions (as comments). *Note that the code continues on the reverse side.*

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

// Complete the class declaration with a private array 'grades' of 11 floats, and
// a public static integer 'pos_4', only one default constructor and destructor,
// which are implemented after declaration (no definitions inside the class):

class Exam {
 public:
  operator std::string();
```

```
// In the constructor body, fill the array 'grades' with possible marks (1.0,2.0,
// 3.0,4.0,5.0,1.3,2.3,3.3,1.7,2.7,3.7) by using a single for loop and a single if
// condition. Initialize 'pos_4' to 0 in the body:
```

```
// In the conversion operator implentation, return the contents of 'grades' as a
// string using std::to_string(), using a range-based loop. Use the std::string
// method find("4.0") to store the position of "4.0" in 'pos_4':

// In the destructor, print out the value of 'pos_4', followed by a new line:

// In the main function, use a decomposition declaration and the above conversion
// operator to print out the elements of myExams, each on a single line:
int main() {
  auto myExams = std::make_tuple( Exam(), Exam(), Exam() );
```

Universität Siegen

In-Class Exercise 2

No pencils, no devices allowed

Advanced Programming in C++ Exercises

Ubiquitous Computing

Student ID: |__|__|__|__|__|__|__|

Your *full* name: _____

Complete below the class declaration, all implementations it promises, and the main function under the respective comments. Make sure that you provide code that will compile in a fully functioning program, and that you adhere to the given instructions (as comments). *Note that the code continues on the reverse side.*

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

// Complete the class declaration with a private array 'types' of 11 doubles,
// and a public static integer 'count', only 1 default constructor and destructor,
// which are implemented after declaration (no definitions inside the class):

class Marks {
 public:
  operator std::string();
```

```
// In the constructor body, fill the array 'types' with all possible marks (1.0,
// 2.0,3.0,4.0,5.0,1.3,2.3,3.3,1.7,2.7,3.7) with a single for loop and a single
// if condition, and increment the 'count' static member:
```

```
// In the conversion operator implentation, return the contents of 'types' as a
// string using std::to_string(), using a range-based loop:

// In the destructor, print out the value of 'count', followed by a new line:

// In the main function, use a decomposition declaration and the above conversion
// operator to print out the elements of myMarks, each on a single line:
int main() {
  auto myMarks = std::make_tuple( Marks(), Marks(), Marks() );
```

Universität Siegen

In-Class Exercise 3

No pencils, no devices allowed

Advanced Programming in C++ Exercises

Ubiquitous Computing

Student ID: |__|__|__|__|__|__|__|

Your *full* name: _____

Complete below the elements of a C++ program under the respective comments. Make sure that you provide code that will compile in a fully functioning program, and that you adhere to the given instructions (as comments). *Note that the code continues on the reverse side.*

```
#include <iostream>

// Create a struct 'Book', which holds a pointer to a title (std::string) and the
// price of a book object. Book's single constructor should initialize these two
// directly through its two parameters, and have an empty body:

// Create a global function 'clone' that takes a pointer to a Book object
// as a sole parameter. It returns a Book object with the same name as the
// parameter's book and 10.0 as the price:

int main() {
  // Create a dynamic array 's' of 3 std::string objects, which is initialized
  // with these strings: "a", "b", and "c"
```

```
    // Create an array 'a' of 3 Book-pointers that is initialized immediately on
    // the same line with 3 pointers to objects. These are constructed with pointers
    // to the contents of the array s above and prices 99, 98, and 97 respectively:









    // Create an array 'b' of three Book objects that is immediately filled by
    // calling the clone function on each of the elements of the array 'a' above:









    // Use a range-based for-loop to print all titles and prices of array b:









    // Ensure that all allocated memory is properly freed:

}
```

Universität Siegen

In-Class Exercise 3

No pencils, no devices allowed

Advanced Programming in C++ Exercises

Ubiquitous Computing

Student ID: |__|__|__|__|__|__|__|

Your *full* name: _____

Complete below the elements of a C++ program under the respective comments. Make sure that you provide code that will compile in a fully functioning program, and that you adhere to the given instructions (as comments). *Note that the code continues on the reverse side.*

```
#include <iostream>

// Create a class 'User', which holds a public pointer to a name (std::string)
// and a public id of a User object. User's single constructor should
// initialize these directly through its two parameters, and have an empty body:
```

```
// Create a global function 'duplicate' that takes a pointer to a User object
// as a sole parameter. It returns a User object with the same name as the
// parameter's user, and 1 as the id:
```

```
int main() {
  // Create a dynamic array 's' of 4 std::string objects, which is initialized
  // with these strings: "1", "2", "3", and "4"
```

```
    // Create an array 'a' of 4 User-pointers that is initialized immediately on
    // the same line with 4 pointers to objects. These are constructed with pointers
    // to the contents of the array s, and id 4, 3, 2, and 1 respectively:









    // Create an array 'b' of four User objects that is immediately filled by
    // calling the duplicate function on each of the elements of the array 'a':









    // Use a range-based for-loop to print all names and ids of array b:

.









    // Ensure that all allocated memory is properly freed:









}
```

Universität Siegen

In-Class Exercise 4

No pencils, no devices allowed

grade: |____|____|

Advanced Programming in C++ Exercises

Ubiquitous Computing

Student ID: |__|__|__|__|__|__|__|

Your *full* name: _____

Complete below the elements of a C++ program under the respective comments. Make sure that you provide code that will compile in a fully functioning program, and that you adhere to the given instructions (as comments). *Note that the code continues on the reverse side.*

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

// Given is the following class to represent a document:
class Document {
  int xPos, yPos;  // document width and height in mm
 public:
  Document(int x, int y = 0) { xPos = x; yPos = y; }
  Document( Document const & d ) = delete;
  Document() = delete;
};

// Declare the class 'Poster' as a child class of Document, with 'title' as its
// only attribute, to hold the poster title. The only Poster constructor should
// call the constructor of Document to initialize the width and height, as well
// as initialize the title, using three parameters, and have an empty body.
// Poster should have a friend method print, which takes a reference to a Poster
// object and does not return anything.
```

```
// Implement the method print as described above, which prints out the title of
// the Poster object. If the title is an empty string, an std::domain_error
// exception should instead be generated with "no title" as a message.
// Hint: string's empty() method does not need a parameter and returns a boolean
```

```
int main() {
  Poster * p[2] = { new Poster(1, 2, "hi"), new Poster(3, 4, "") };
  // use a ranged-based loop to pass the above 2 posters to print(), and handle
  // any exceptions that might be generated by print here:
}
```

Universität Siegen

In-Class Exercise 4

No pencils, no devices allowed

grade: |____|____|

Advanced Programming in C++ Exercises

Ubiquitous Computing

Student ID: |__|__|__|__|__|__|__|

Your *full* name: _____

Complete below the elements of a C++ program under the respective comments. Make sure that you provide code that will compile in a fully functioning program, and that you adhere to the given instructions (as comments). *Note that the code continues on the reverse side.*

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

// Given is the following class to represent a user interface window:
class Window {
  uint16_t winWidth, winHeight;  // the window width and height in pixels
 public:
  Window() = delete;
  Window( Window const & w ) = delete;
  Window( uint16_t w, uint16_t h = 0 ) { winWidth = w; winHeight = h; }
};

// Declare the class 'PopupWindow' as a child class of Window, with 'header' as
// its only attribute, to hold the window header title. Its only constructor
// calls the constructor of Window to initialize the width and height, as well
// as initialize the header, using three parameters, and have an empty body.
// PopupWindow should have a friend method print, which takes a reference to a
// PopupWindow object and does not return anything.
```

```
// Implement the method print as described above, which prints out the header
// title of the PopupWindow. If its header is an empty string, an std::length_error
// exception should instead be generated with "no header" as a message.
// Hint: string's empty() method does not need a parameter and returns a boolean
```

```
int main() {
  PopupWindow * p[2] = { new PopupWindow(1, 2, "hi"), new PopupWindow(3, 4, "") };
  // use a ranged-based loop to pass the above 2 popup windows to print(), and
  // handle any exceptions that might be generated by print here:
}
```

Universität Siegen

In-Class Exercise 5

No pencils, no devices allowed

grade: |____|____|

Advanced Programming in C++ Exercises

Ubiquitous Computing

Student ID: |__|__|__|__|__|__|__|

Your *full* name: _____

Complete below the elements of a C++ program under the respective comments. Make sure that you provide code that will compile in a fully functioning program, and that you adhere to the given instructions (as comments). *Note that the code continues on the reverse side.*

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

class User {  // class representing a server's user
  // Declare below the infoString const method so that it fullfills the Non-Virtual
  // Interface Idiom. In User, this method should return username and name.

 protected:
  std::string name, userName;  // the full name and the user name of the user
 public:
  User(std::string f, std::string u) { name=f; userName=u; }
  void info() { std::cout << infoString() << '\n'; }
};

/* Create two subclasses 'Admin' and 'Guest' of 'User' below. Admin has a private
   attribute 'level', Guest has a private attribute 'loginTime', both of the type
   std::string. Both should have one sole constructor to take three parameters for
   initializing name, userName, and their attributes, by using User's constructor
   and avoiding statements in the constructor's body. Implement methods for both
   classes using the Non-Virtual Interface idiom, that print out the class name and
   value of the private attribute when the object's info() method is called. */
```

```
/* Show in main() below, on an example array, that objects of all the above three
   classes use polymorphism when displaying the object's details with info(). */

int main() {

}
```

Universität Siegen

In-Class Exercise 5

No pencils, no devices allowed

grade: |____|____|

Advanced Programming in C++ Exercises

Ubiquitous Computing

Student ID: |__|__|__|__|__|__|__|

Your *full* name: _____

Complete below the elements of a C++ program under the respective comments. Make sure that you provide code that will compile in a fully functioning program, and that you adhere to the given instructions (as comments). *Note that the code continues on the reverse side.*

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

class File {  // class representing a single data file
  // Declare the fileDetails const method so that it fullfills the Non-Virtual
  // Interface Idiom. This method should, for File, return the name and extension.
```

_____

_____

_____

```
 protected:
  std::string name, ext;  // the file name and extension
 public:
  File(std::string n, std::string e) { name=n; ext=e; }
  void details() { std::cout << "Details: " << fileDetails() << "\n"; }
};

/* Create two subclasses 'Icon' and 'Text' of File below. Icon has a private
   attribute 'type', Text has a private attribute 'encoding', both of type
   std::string. Both should have one sole constructor to take three parameters for
   initializing name, extension, and their attributes, by using File's constructor
   and avoiding statements in the constructor's body. Implement for both classes
   methods using the Non-Virtual Interface idiom, printing out the class name and
   value of the private attribute when the object's details() method is called.*/
```

_____

_____

_____

_____

_____

_____

_____

_____

_____

```
/* Show in main(), on an example array, that objects of all above three classes
   use polymorphism when displaying the object's details with details(). */

int main() {

}
```