Summer Term
2020

Prof. Harald Köstler
Jan Hönig
Dominik Thönnes

Exam in

Advanced Programming Techniques

June 8, 2020

Name: ____________________

Date of birth: ____________________

Matriculation number: ____________________

Course of studies: ____________________

Please do not fill out anything below this line!

Total number of points: ______ of 60

Grade: ________

Passed: ☐ Yes / ☐ No

Problem 1: Definitions and Terms (17 points)

(a) (7P) List the following:

One class from the random library: ____________________

Two associative containers: ____________________

One sequential container: ____________________

Three C++ keywords for explicit casts: ____________________

(b) (3P) List three different classes implemented in the C++ standard library and in which header file they can be found.

(c) (4P) Given is the function `fct` which takes a reference to a `double` and a pointer to a constant `std::list` as input and returns nothing.
Using the `std::function` library type, define a variable `f` and initialize it with `fct`.

(d) (3P) In which namespace is `steady_clock::now()` defined, and why do you need to access `now()` via the scope operator?

Problem 2: Errors and Function Matching (12 points)

Fix all compiler errors in the following incorrect C++ program.

```
#include <string>
#include <cstddef>

class HasPtr {

  HasPtr(const std::string &s = std::string()):
   ps(new std::string(s)), i(0), use(new std::size_t(1)) {}

  HasPtr(const HasPtr &p): ps(p.ps), i(p.i), use(p.use) { ++*use; }

HasPtr& operator=(const HasPtr&) const;

~HasPtr();

private:
    std::string &ps;
    int    i;
  std::size_t *use;
};

HasPtr::~HasPtr()
{
if (--*use == 0) {
delete [] ps;
delete [] use;
}
return 0;
}

HasPtr& HasPtr::operator=(const HasPtr &rhs) const
{
++*rhs.use;
if (--*use == 0) {
delete ps;
delete use;
}
ps = rhs.ps;
i = rhs.i;
use = rhs.use;
return *this;
}

int main()
{
HasPtr h(2);
}
```

Problem 3: Programming with the Standard Library (8 points)

Please note: The questions assume that all necessary header files from the Standard Library are included and an implicit `using namespace std;`*. Likewise, you can safely assume the same for your code!*

(a) (4P) Implement the function

```
template<typename KEY, typename VALUE>
void deleteValue(std::map<KEY, VALUE> & m, VALUE v)
```

- that removes the first key-value pair where the value is equal to `v`

Example usage:

```
map<char,int> m = {{'a',1},{'b',3},{'c',4}};
deleteValue(m,3);
cout << m << " ";
```

Expected output of the snippet above (if `operator<<` is overloaded):

```
a: 1
c: 4
```

(b) (4P) Implement the function

```
template<typename T>
int countDuplicates(std::vector<T> const & one, std::vector<T> const & two)
```

- that counts all elements that are in both `one` and in `two`
- use the STL alorithms library to find duplicates
- you can assume that there are no duplicates within one vector

Example usage:

```
std::vector<int> one {1,2,3,4,5,6};
std::vector<int> two {5,6,7,8,9};
cout << countDuplicates<int>(one,two) << endl;
```

Expected output of the snippet above:

```
2
```

since 5 and 6 are contained in both vectors

Problem 4: Classes (11 points)

Please note: The questions assume that all necessary header files from the Standard Library are included and an implicit `using namespace std;`*. Likewise, you can safely assume the same for your code!*

Write a class "IntList" such that following main function works:

```
IntList list1;
IntList list2(2);

list1 += 9;
list2 += 4;
list2 += 5;

std::cout << list1 << std::endl;
std::cout << list2 << std::endl;
std::cout << list2.elementSum() << std::endl;
```

and has the following output:

```
9,
2,4,5,
11
```

Make sure to not expose the internal data structure! Don't forget to implement the `operator<<` function given on the next page!

```
class IntList {
```

```
std::ostream & operator <<(std::ostream &os, const IntList & list){
```

Problem 5: Short Questions (12 points)

(a) (4P) What is the output of the following program and why?

```
#include<iostream>
using namespace std;

int &f(int x) {
  return x;
}
int main() {
  int x = 5;
  f(x) = 10;
  cout<<x;
}
```

(b) (2P) What is a pure virtual function?

(c) (2P) How is static and dynamic polymorphism realized in C++?

(d) (2P) What does the std::allocator class do?

(e) (2P) If the 'new' operator is to be avoided, what is a reasonable solution?

Problem 6: Project (max. 12 bonus points)

(a) Which race did you implement in your group?

(b) What is your group's name?