Programming in C++ II
Jacobs University Bremen
Dr. Kinga Lipskoch

Course: CH08-320143
March 19th, 2019

Practice Sheet

First Name: .....

Last Name: .....

Matriculation Number: .....

- Read all the following points before proceeding to the solution.
- Write immediately your name on this sheet.
- Write clearly. Take into consideration that C++ is a case sensitive language.
- Indent your code in a sensible way.
- Books, slides, notes or other documents are not allowed.
- If you need more space to solve the exercises you may use also the back of each page.
- Read carefully the questions and strictly adhere to the requirements.
- You have two hours to solve this test.
- Any attempt to cheat leads to an immediate fail.
- By signing this sheet you imply you read and understood all of the above.

Signature: .....

%	0.00 - 39.49	39.50 - 44.49	44.50 - 49.49	49.50 - 54.49
Grade	5.0	4.7	4.3	4.0

%	54.50 - 59.49	59.50 - 64.49	64.50 - 69.49	69.50 - 74.49
Grade	3.7	3.3	3.0	2.7

74.50 - 79.49	79.50 - 84.49	84.50 - 89.49	89.50 - 94.49	94.50 - 100.00
2.3	2.0	1.7	1.3	1.0

Reference Constructors, Methods and Functions (Selection)

```
ifstream(const char* filename, ios_base::openmode mode = ios_base::in);

ofstream(const char* filename, ios_base::openmode mode = ios_base::out);

void push_back(const T& el)

void push_front(const T& el)

void pop_front()

void pop_back()

T& back()

T& front()

bool empty()

size_type size() const;

size_type count(const key_type& k) const;

iterator find(const key_type& k);

const_iterator find(const key_type& k) const;

pair<iterator,bool> insert(const value_type& val);

template <class ForwardIterator, class T>
void replace(ForwardIterator first, ForwardIterator last,
             const T& old_value, const T& new_value);

logic_error(const string& what_arg);

logic_error(const char* what_arg);

exception() throw();

const char* what() const throw();
```

Problem P.1 *Summation of an array* (3 points)

Write a template function to complete the program given below. Do not forget appropriate header files and other statements to have a fully functional program.

```
int main(void) {
    int inum[5] = { 10, 20, 30, 40, 50 };
    double dnum[5] = { 1.1, 2.2, 3.3, 4.4, 5.5 };

    cout << "Sum of inum is: " << sumArray(inum, 5) << endl;
    cout << "Sum of dnum is: " << sumArray(dnum, 5) << endl;
}
```

Problem P.2 *A Queue class* (3+3 = 6 points)

Consider the following class:

```
template<class T>
class Queue {
    private:
        QueueItem<T> *front;
        QueueItem<T> *back;
    public:
        Queue();
        ~Queue();
        T remove(int pos);
        void add (const T&);
        bool is_empty() const {
            return front == NULL;
        }
};
```

a) Write the implementation of a destructor of this class, which empties the queue of items. You can assume that all other methods have been implemented and are usable.

b) Also write an example program that creates a queue with 10 integer entries.

Problem P.3 *A Deque* (3 points)

Write a program that stores all multiples of 5 from 5 to 500 into a `deque`. Then print all numbers on the screen using a constant iterator.

Problem P.4 *Sorting strings* (4 points)

Write a program which reads in strings into a `list` until `ZZ` is entered (`ZZ` is an end-marker and not part of the data). Then print all strings which have been entered in alphabetical order.
You must use the STL to store the strings into a list and must use an iterator to print all the elements of the list. The container `list` provides a method `sort()` to sort elements of a container.

Problem P.5 *An implementation of* `vector` (4 points)

Consider the following code:

```
#include <vector>
class Building {
    public:
        virtual int price() = 0;
};
class Neighborhood {
    public:
        std::vector<Building*> coll;
    public:
        /* constructors, destructors, getter and setter methods omitted */
        double overallprice();
};
```

Write the definition of the `overallprice` method. The method returns the sum of the prices of the buildings stored in the vector `coll`. Assume the vector is completely filled. Consider the presence of the overloaded operator `[]` to access elements in vector objects, and of the member function `size()` which returns the number of elements in the vector.

Problem P.6 *A worker class* (4 points)

a) Write an appropriate class for the program below such that the class declaration, definition and this `main` function will compile and run together.

b) Also complete the missing code in the `main` function, so the workers are written to a file named "`list.dat`".

```
int main() {
  worker a(234, "John McEnroe");
  worker b(324, "Jack Nicholson");

  cout << a << b;

  cout << "Dumping to file...: " << endl;
  ...
  ...
  ...
  ...
  ...
  ...
  ...
  return 0;
}
```

Problem P.7 *Multiple inheritance* (2 points)

You will either get a full program (without any compilation or other errors) and you will have to write down the exact output, or get a program with missing parts which you have to add such that a given `main` function will compile and run.

Problem P.8 *Write and use exception class* (2 points)

Write an exception class derived from the class `logic_error`. Write a constructor and override the `what()` method. Then write a `main` function where should catch the previously defined exception type.

Problem P.9 *Unit tests for the `Complex` class* (4 points)

Consider one of the `Complex` class implementation with overloaded operators. Write simple unit tests for all functionality of the class. Then group the tests into a few different categories, and write and use a Makefile for compiling the running the tests separately and all together.