_	

Name Matr.-Nr.

Exam Programming III – CTS

WS 17

Prof. Dr. K. Baer

Auxiliary materials: 1 page DIN A4, double sided

Preparation time: 90 Min.

- 1. Please enter name and matriculation number first!
- 2. Check the completeness of the task sheets.
- 3. The exam consists of 5 tasks. Get a quick overview of the tasks and get started on the task that will most likely bring you a sense of achievement.
- 4. Read the task carefully before you try to solve the task!
- 5. Use the space provided on the task sheets to answer the questions.
- 6. Write legibly. Unreadable parts are rated 0 points!

Good Luck!

Task	1	2	3	4	5	Sum
Points	23	8	16	10	33	90
achieved						

Task 1 (23 Points = 13+2+2+2+4)

The following class definition shall be given:

```
1
    class Part {
2
    public:
3
         Part()
                              { cout << " cPart"; }
         Part(const Part& a) { cout << " copyPart"; }</pre>
4
                             { cout << " ~Part"; }
5
         ~Part()
6
    };
7
8
    class Base {
    private:
9
10
         Part p;
11
    public:
12
                              { cout << " cBase"; }
        Base()
        Base(const Base& b) { cout << " copyBase"; }</pre>
13
14
        ~Base()
                               { cout << " ~Base"; }
15
16
        void method1(Base b) { cout << " m1Base"; }</pre>
17
    };
18
19
    class Child : public Base {
20
    private:
21
        Part* ptrP;
22
    public:
23
                               { cout << " cChild"; ptrP=0;}
        Child()
        Child(const Child& c) { cout << " copyChild"; ptrP=c.ptrP; }</pre>
24
25
        ~Child()
                               { cout << " ~Child"; if (ptrP) delete ptrP;}
26
        void method1(Base b) { cout << " m1Child"; }</pre>
27
        void method1(Base* b) { cout << " m1_Child"; b->method1(*b); }
28
                         { cout << " m2Child"; ptrP = new Part(); }
29
        void method2()
30 };
```

a) For the test () function below, after each line, specify which outputs appear on the console during evaluation.

Please write the word "NOTHING" in cases where you want to express that no no output is generated. Leaving a field blank means that you have not provided an answer and therefore will not get points for it.

No	Code	Output (write NOTHING if no outputis generated)
1	void test(){	
2	Child c1;	cPart cBase cChild
3	Child c2 = c1;	cPart cBase copyChild
4	Base b1;	cPart cBase
5	Base* ptrB = &c2	NICHTS
6	ptrB->method1(c1);	cPart copyBase m1Base ~Base ~Part
7	ptrB = new Child();	cPart cBase cChild
8	static_cast <child*> (ptrB)→method2();</child*>	m2Child cPart
9	c1.method2();	m2Child cPart
10	delete ptrB;	~Base ~Part
11	Child* ptrC = &c1	NICHTS
12	ptrC->method1(&c2);	m1_Child cPart copyBase m1Base ~Base ~Part
13	delete ptrC;	~Child ~Base ~Part
14	}	~Base ~Part ~Child ~Base ~Part ~Child ~Base ~Part

b) Name the difference between overloading and overriding methods. Which lines in the class definition of page 2 contain Overloading and which ones Overriding?

Methods with the same name:

Overload: method of the same name in the same class with different parameter list (lines 27 & 28)

Override: Method with the same name in different classes of an inheritance hierarchy with the same parameter list. (lines 16 & 27)

 c) In row 10 and 21 of the class definitions (see page 2) different types of relationship between classes are used.
 How do you call these two different types of relationships?
 What are the consequences using these two types of modeling?

10: Composition → Lifetime dependency

21: Aggregation → Memory management must be solved, otherwise there is a risk of memory leakage

d) Where does a memory leak occur in method test ()? Explain why!

In line 8 Part-Object is created, which is not deleted again in line 10 when deleting prtB, because Destructor is not virtual.

In line 9 a new Part object is created, the old one creates a memory leak.

e) Now, the destructor of class *Base* as well as the method *method1* shall be declared as virtual.

Specify the changes in the output of test().

6: cPart copyBase m1Child ~Base ~Part

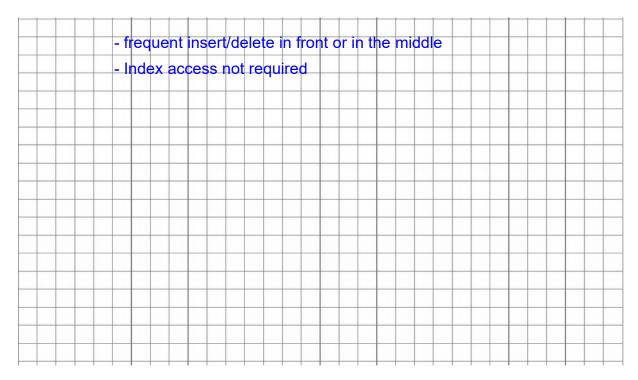
8: m1_Child cPart copyBase m1Child ~Base ~Part

10: ~Child ~Part ~Base ~Part

12: m1_Child cPart copyBase m1Child ~Base ~Part

Task 2 (8 points = 2 + 6)

a) In which cases should a programmer prefer std :: list <T> to
 std :: vector <T> and vice versa?



b) What problems may arise when applying explicit casts to derived classes?

dow	ncas	t: exi	olici	t type	cas	st ir	ı cla	ass	hie	era	rch	v d	ow	nwa	ards	S .					
	sible																is c	of ba	ase	clas	SS
type						- 22							_	Ĭ	7						Γ
	mple																				Г
	Obj*		eo :	= ne\	v Ge	оО	bi(ł	(00	rd(1.2	2)):	6 6									Γ
	le* pt																				Г
						2															Γ
ntrC	ircle	- etc	tic	cast	circ	.*ما	>(nt	rC	20)	. //	val	idl									
puc	ircle-	- Ste		Cast	CIIC	//	dro.		of Of	oir	vai	iu:									
puc	ircle-	ula	νν(),		۸. ا	"	Jia	V()	OI ofit	CIIC	i C										Г
puc	il Cle-	→Cai	CUII	Cum	(),	// -		Hue	3111	lec											
dura	omio	000											_		_						L
	amic_						-1-*		1		\ \ /	// N I		D4						-	
pu C	ircle	= qy	nan		ası<	CIT	Jie	≯ (p	uG	Эес), [/ IN	uII-	Pu!							
іт (р	trCirc							nali	ng												L
	els	se	nor	mal	oroc	ess	ıng														
																					П

Task 3 (16 Points=4+6+6)

A game requires a class Player (see below), which essentially stores the score points and the time required. The players are managed in a scorelist of type vector <Player> (see test () - method).

Elementary functions in a game are:

- find the best player or
- find Players who are better or worse than a certain player.

```
class Player{
private:
      std::string name;
      int points;
      int time;
public:
      Player(std::string name, int p, int t) : name(name), points(p),
                                                 time(t){}
      int getPoints() const { return points; }
      int getTime() const { return time; }
      std::string toString() const {
            std::stringstream buffer;
            buffer << "Name: " << name << ", Points: " << points << ",</pre>
                       Time: " << time;
            return buffer.str();
      }
      bool operator>(const Player& other) const {
            return this->points > other.points;
      }
      bool operator<(const Player& other) const {</pre>
            return this->points < other.points;</pre>
      }
};
std::ostream& operator<<(std::ostream& os, const Player& player){</pre>
      return os << player.toString();</pre>
}
void test(){
  std::vector<Player> scorelist;
  std::vector<Player> top;
  Player* Red = new Player("Red", 10, 15);
  Player* Purple = new Player("Purple", 20, 25);
  Player* Blue = new Player("Blue", 30, 25);
  Player* Yellow = new Player("Yellow", 40, 40);
```

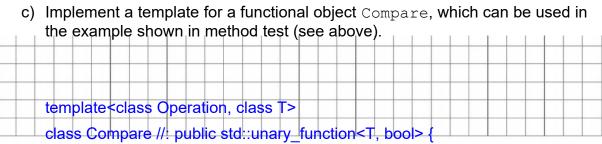
```
scorelist.push_back(*Red);
  scorelist.push back(*Purple);
  scorelist.push back(*Blue);
  scorelist.push_back(*Yellow);
  typedef std::vector<Player>::iterator Iter;
  std::ostream_iteratorkPlayer>Output(std::cout, "\n");
  Iter start = scorelist.begin();
  Iter end = scorelist.end();
  Iter max = std::max_element( start, end,
                                              CompareScore() ) ;
                                                                    // TODO
  std::cout << (*max) <k std::endl;</pre>
  my copy if (start, end, std::back inserter(top),
                                                                    // TODO
                   Compare<std::greater<Player>, Player&>
                                                                    // TODO
                                       (std::greater<Player>(),
                                                                 *Purple) );
  copy(start, end, Output);
  copy(top.begin(), top.end(), Output);
You can use the STL function max element to determine the largest element, if
you have a corresponding comparison function.
   a) Implement a functional object CompareScore, which compares players
      according to the formula: 2 * points / time.
      struct CompareScore{
            bool operator()(const Player& p1, const Player& p2){
                  return (2*p1.getPoints()/(p1.getTime())) < (2*p2.getPoints()/
                                                              (p2.getTime()));
      };
```

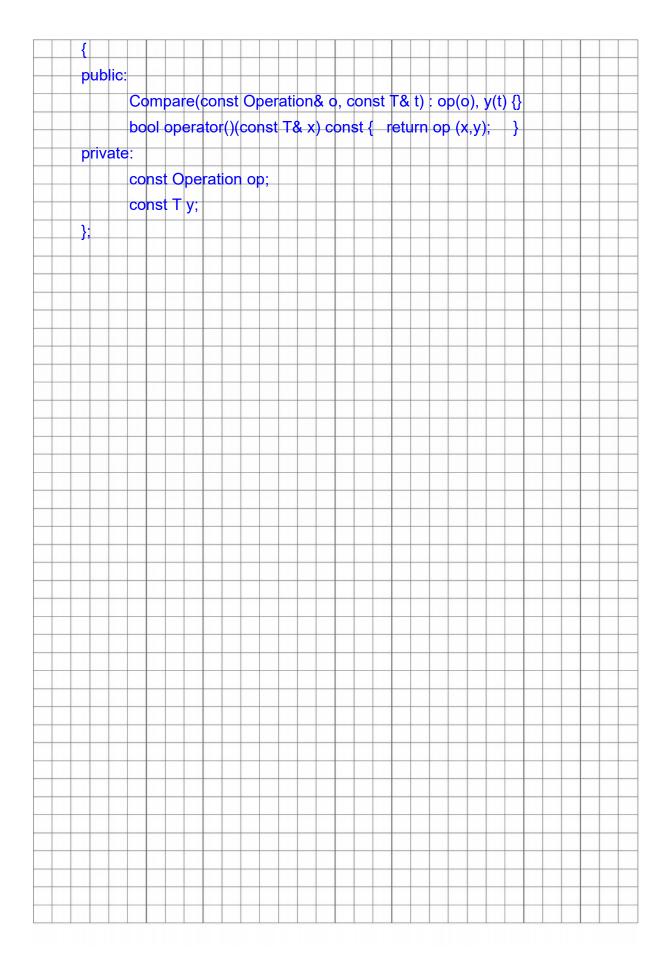
b) With my_copy_if, all elements (players in the test-method above) that satisfy a condition (in the example above, those who are better than the player Purple) should be copied to a result container. Let us assume, that the function copy_if was forgotten in the STL, so this function has to be implemented.

As a parameter it requires:

• two iterators on the area in the source container from which values are

	0		ite							e c tior					ult	CO	nta	aine	-r	to	wh	ich) th	le.	res	ult	s w	ill	he
			red		01 1		LITC	P	701					00	ait	00	1100	4111	JI,	ıo	VV 1	1101				ait	J 44	***	
	0		oina		fur	nct	ior	ı fc	r c	on	าตล	arir	a t	he	va	llue	es.												
	0		turr																r tl	ne	las	t ir	ıse	rte	d e	ele	me	nt	in
			re																										
	te	mpla	ate	<cl< td=""><td>ass</td><td>: Ir</td><td>ılte</td><td>r,</td><td>cla</td><td>SS</td><td>Οι</td><td>ıtlt</td><td>er,</td><td>cla</td><td>ISS</td><td>U</td><td>nΟ</td><td>p></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></cl<>	ass	: Ir	ılte	r,	cla	SS	Οι	ıtlt	er,	cla	ISS	U	nΟ	p>											
		- [ľ 1			l n	\sim		1					
		utIte	\rightarrow	-	-	_				-	_	-			ol,	Ou	uu	יוב	168	١, ا	UII	Οþ	Οļ	ሃለ					
	_		for	(In	Ite	r i	= f	irst	; i	!=	las	t; -	+i	\{															
	_		\square		-if	(0	p(*	i)).	_																				
							P (4:																_	
	-		\vdash		-	_		"d	es	[++		*i;					_						_					-	_
	-	-			-}	_											-						_						_
			3			-																	> 1						
	+		1			-											_		_				_						
			ret	urr	ı d	es	,																						
	}																												
																	100												
	_		\square			_				_																			
										8 2										7									
_	-	-	\vdash			_																							
	-		\vdash			_														7 - 9									
-	-	-				_																							
						_											7. 7.												
_					-	_											-												





Task 4 (10 points=4+3+3)

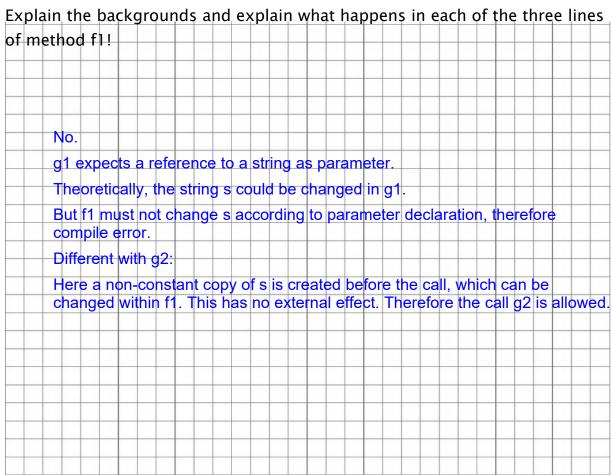
a.) The following program code shall be given:

```
#include <iostream>
void g1(std::string& s) { std::cout << s << std::endl; }
void g2(std::string* sptr) { std::cout << *sptr << std::endl; }

void f1(const std::string& s) {
   g1(s);
   std::string localCopy = s;
   g2( &localCopy );
}

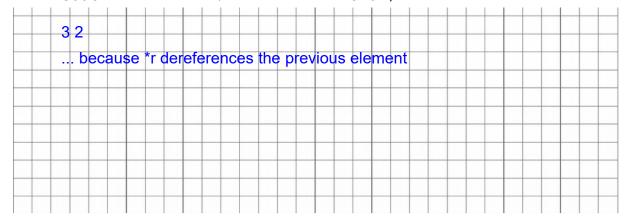
int main() {
   f1("Hallo");
   return 0;
}</pre>
```

Is the program compilable?



b.) What is the output of the following code fragment? The values of *i and *r do not match. Why?

```
const int f[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
vector<int> v(f, f+10);
vector<int>::iterator i = v.begin() + 3;
vector<int>::reverse_iterator r(i);
cout << *i << " != " << *r << endl;</pre>
```



c.) The following program will not be compiled. Why?

```
const int f[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
vector<int> v;
copy(f, f +10, front inserter(v));
```

Vector --> push_front not available

front inserter cannot construct a corresponding inserter

Task 5 (33 points = 6+8+4+5+6+4)

Implement (without the use of the class vector of the standard template library) a new class Point, which should simplify the handling of coordinates. When creating an object of this class, it shall be possible to specify, how many dimensions are available.

Example:

```
Point p(1); // 1-dimensional, only a x-component
Point q(2); // 2-dimensional, q has a x and a y-component
Point r(3); // 3-dimensional, r has x,y and z-components
```

In order to be able to keep the dimension arbitrary, the components should be stored in an array which is requested at the time of object creation. The data type of the components is double.

- a) Implement a class Point that offers this
- b) In addition, access to a component shall be possible like the access to a single element of an array.

Examples:

```
p[0] // x-component of point p q[1] // y-component of point q r[2] // z-component of point q
```

If an attempt is made to access a component outside the dimension, the exception "OutOfDimension" shall be generated.

The above component access mechanism should also allow the component values to be set or read so that, for example, the following accesses are possible:

c) The construction of a 'point object' from another 'point object' shall be possible.

Example:

```
Point q(3);
Point p = q;
```

d) The assignment of a point object to a (possibly different) point object only shall be possible if the dimension of the expression on the right side of the

assignment is less than or equal to the dimension of the object on the left side. If the dimension of the object on the right side is really smaller, only the existing components will be copied.

In the event of an error, the exception "DimensionMismatch" should be generated.

- e) Develop an output operator for Point-objects. The output produced should look like: (100 5 42). The data encapsulation should not be affected by this!
- f) In addition, develop a simple test program, which only has the task of creating and catching the two exceptions <code>DimensionMismatch</code> and <code>OutOfDimension</code> when using class <code>Point</code>. It is sufficient if the test program merely outputs an error message on the screen as error handling.

Hint:

The class is to be fully implemented, i.e. with all required constructors, destructors, etc. and possibly additional classes have to be added.



```
private:
 double* components;
 int len;
                                                                      // 2 P
public:
 Point(int length):len(length){
  components = new double[len];
      // 2 P
 Point(const Point& other){
  len = other.len;
  components = new double[len];
  for(int i = 0; i< len; ++i){
   components[i] = other.components[i];
      // 4 P
 virtual ~Point(){
  if (components) {
   delete[] components;
      // 2 P
 Point& operator=(const Point& other){
                                                        // 5 P
      if( this == &other){
             return *this;
  if(other.len > this->len){
   throw Error("Dimension mismatch ");
  }
  else{
   for(int i = 0; i < other.len; ++i){}
    components[i] = other.components[i];
```

