2020/11/30
CS Department
Dr. Klaus Baer

Bar Code

Programming III,
Fall 2020
Mid Term Exam

Instructions: **Read Carefully Before Proceeding.**

1- No calculators, book or other aids are permitted for this test.

2- Write your solutions in the space provided. If you need more space, write on the back of the sheet containing the problem.

3- Attempt as much of the problems as you can within the time limits. The more you solve the higher your score is expected to be.

4- Read all the problems carefully before starting, and start with the easiest question you find.

5- This exam booklet contains 16 pages, including this one. *Extra sheets of scratch paper are attached* and have to be kept attached.

6- When you are told that time is up, stop working on the test.

7- Total time allowed for this exam is ***120 min.***

Good Luck!

Question Number	1	2	3	4	5	6	-	-	-	-	Total
Maximum Score	12	28	10	16	8	26					100
Obtained Score											

Task 1 (12 Punkte = 5 + 4 + 3)

a) The following program fragment shall be given. Indicate which assignments lead to a compile error and justify your answer.

```
int main() {
      char c1 = 'A';
      char * p1 = &c1;
      const char * p2 = &c1;
      char * const p3 = &c1;
      const char * const p4 = &c1;
```

	Valid?	Justification
`*p2 = *p1`	no	writing forbidden because pointer to constant
`p2 = p1;`	ok	Read-Only pointer can be changed
`p3 = p1;`	No	writing forbidden constant pointer
`p1 = p3;`	ok	P1 may be changed
`p1 = p2;`	No	Conversion from 'const char *' to 'char *' not possible.

```
}
```

b) Explain the various meanings of `const` in C++:

- `const double pi = 3.14156;`

  declaring a constant value

- `void foo(const MyClass &p){}`

  method foo promises to its users that it will not change the value of

  reference p. Similar to call-by value without copying

- `void member_foo() const {}`

  method member_foo will not change the status of the object it is called for. It can be called within foo in the last point: p.member_foo() does not change p.

- `const MyClass & foo(){}`

  foo() returns a const reference to MyClass. Via this reference, the value may not be changed

c) What does "this" mean in C++ and when is it used? Give an example.

this is automatically defined within methods as a constant pointer to the object itself for which the method was called

Usage: to uniquely reference elements of the current object

Task 2 (28 Points = 14+2+4+4+4)

The following class definition shall be given:

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

a) For the test () function below, after each line, specify which outputs appears on the console during evaluation.
Please write the word "NOTHING" in cases where you want to express that no output is generated. Leaving a field blank means that you have not provided an answer and therefore will not get points for it.

	Code	Output (write NOTHING if no output is 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();`	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 of the same name:

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

Overwrite: Method of 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? Explain!

10: Composition → Lifetime dependency
21: Aggregation → Memory management must be clarified, otherwise memory leakage is likely to occur

d) Where does a memory leak occur in method test ()? Explain why!
In line 8 a part object is created, which is not deleted in line 10 when prtB is deleted, because destructor is not virtual.
In line 9 a new Part object is created, the old one forms 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 3 (10 Points = 5 + 5)

a) Please answer the following questions::

	Correct	Wrong
A static data element must be defined and initialized separately	x	
Static data elements of a class declared as static occupy memory space even if no object of the class has been created	x	
Static data elements of a class are always public elements		x
Static data elements of a class are always const elements		x
In static methods the "this" pointer is available to access other elements of the current object.		x

b) What is a **namespace** in C++?
Explain the meaning and purpose of namespace and use a syntax example to explain how to use namespaces.

Namespaces provide a method for preventing name conflicts in large projects.
Symbols declared inside a namespace block are placed in a named scope that prevents them from being mistaken for identically-named symbols in other scopes.
Multiple namespace blocks with the same name are allowed. All declarations within those blocks are declared in the named scope.

```
namespace Geo {
        class GeoObject{
                ...
        };
} // end namespace
```

Or

```
Using namespace std;
```

Task 4 (16 Punkte)

Develop a `merge()` function that merges two sorted int-arrays and stores the sorted result in a third array.

The function expects two sorted arrays, their lengths and the third array as arguments. After merging, it returns the number of elements in the third array as return value or -1 if one of the two arrays to be merged is not sorted in ascending order.

a) Develop a function `isSorted()` which checks whether a given array is sorted in ascending order. The function takes the array to be checked and its length as parameters and returns a boolean value indicating the result.

```
bool isSorted( int arr[], int len ){
for (int *p = arr; p < arr + len-1; ++p ){
if ( *p > *(p+1) )
return false;
}
return true;
}
```

b) Develop the `merge()` function

```
int merge( int arr1[], int arr2[], int len1, int len2, int dest[]){
int *p1 = arr1;
int *p2 = arr2;
int *p3 = dest;

if ( !isSorted(arr1, len1) || !isSorted(arr2, len2) ){
return -1;
}

for ( ; p1 < arr1+len1 && p2 < arr2+len2; ++p3 ){
if ( *p1 <= *p2){
*p3 = *p1;
++p1;
}
else {
*p3 = *p2;
++p2;
}
}

while ( p1 < arr1 + len1){
*p3++ = *p1++;
}

while ( p2 < arr2 + len2){
*p3++ = *p2++;
}
return (p3 - dest);
}
```

Task 5 (8 Points = 4 + 4)

a) What does `inline` mean and what is it used for?
How can methods be declared inline?
When should you declare methods inline?

Inline avoids overhead when calling a function
It is an advice to the compiler to substitute each call to the inline method with the machine code. This enlarges the executable but it is more efficient in terms or runtime as no function call has to be made (safe return address, create parameters on stack, jump to method, clean stack, save return value on stack, jump back to return address)

Can be declared explicit with keyword “inline” in front of the method definition
Can be implicitly declared inline by implementing the method within the class declaration.

As it enlarges the executable, use it for small functions which are called often. Makes no sense in recursive programming, as in this case the compiler may not follow the advice.

b) Explain the term Polymorphism in context of object oriented programming in C++.

Core concept of oop.
The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.
Objects of different types can be accessed through the same interface.
Each type can provide its own, independent implementation of this interface.
It is achieved by function overriding.
occurs when a derived class has a definition for one of the member functions of the base class. That base function is said to be overridden.
In C++, such a method has to be declared “virtual” in the base class. In this case, the overridden function will be bound at runtime.

Task 6 (26 Points = 6+2+2+3+3+4+5+1)

Define a class `Parable`, to display parables. A parable like

$$y = a \cdot x^2 + b \cdot x + c$$

is described by three double values a, b and c. They shall be represented internally by a dynamic array (so that theoretically your implementation could also serve as a basis for higher-order polynomials). The class `Parable` manages a pointer of type double called `data` as a private component. This pointer will point to an array of length 3 of type double. The first entry in this array is the value `a`, second entry is the value `b` and third entry is the value `c`.

The class `Parable` has the following public member functions:

```
double evaluate(double);

int determineZeroCrossings(double&, double& );
```

Additionally the class has a private element function to determine if zeros exist:

```
int getNoOfZeroCrossings(void)
```

As a result, it returns the number of existing zero crossings 0, 1 or 2.

a) Specify the class declaration:

```
class Parabel {
private:
double* daten;
public:
Parabel(double a, double b, double c);
Parabel(const Parabel&);
~Parabel( );

double auswerten(double);
int bestimmeNullstelle(double&, double& );
private:
int anzahlNullstellen(void);
};
```

b) Implement the constructor.
It should allow the specification of the three coefficients a, b, c.

```
Parabel::Parabel(double a, double b, double c){
daten = new double[3];
daten[0]=a;
daten[1]=b;
daten[2]=c;
}
```

c) Implement the destructor.

```
Parabel::~Parabel( ){
delete[ ] daten;
}
```

d) Implement the copy constructor.

```
Parabel::Parabel(const Parabel& other){
daten = new double[3];
this->daten[0] = other.daten[0];
this->daten[1] = other.daten[1];
this->daten[2] = other.daten[2];
}
```

e) Explain why the class `Parable` needs a copy constructor.

In the constructor memory is reserved dynamically for an array. It must be ensured in the copy constructor that memory is also reserved in the copied object. Otherwise, copy and original would work on the same array and mutually overwrite each other's data, since the default constructor generated by the system would only create a so-called "flat copy", i.e., only a copy of the pointer.

f) Implement the method `int getNoOfZeroCrossings(void)`. It evaluates the discriminant:

$$D = b^2 - 4 \cdot a \cdot c$$

If the discriminant has a value < 0, the corresponding parable has no zero crossings. If the value = 0 it has one and if the value > 0 has two zero crossings.

Pay attention to special features in your implementation when comparing two double values!!

```
int Parabel::anzahlNullstellen(){
double a = daten[0];
double b = daten[1];
double c = daten[2];
const double genauigkeit = 1.0e-10;

double diskriminante = (b * b ) - (4 * a * c);
```

```
int anzahlNullstellen = 0;

if ( abs(diskriminante) < genauigkeit){
anzahlNullstellen = 1;
}
if ( diskriminante > genauigkeit ){
anzahlNullstellen = 2;
}

return anzahlNullstellen;
}
```

g) Implement the method:

```
int determineZeroCrossings(double&, double&)
```

It is supposed to return the number of zeros, while the zeros themselves are returned by references.

```
int Parabel::bestimmeNullstelle(double& n1, double& n2){

int anzahlNullstellen = this->anzahlNullstellen();

if (anzahlNullstellen > 0){
double a = daten[0];
double b = daten[1];
double c = daten[2];

if (anzahlNullstellen == 1){
n1 = -b / (2 * a);
}
else {
double diskriminante = (b * b ) - (4 * a * c);
n1 = (-b + sqrt(diskriminante)) / (2 * a);
n2 = (-b - sqrt(diskriminante)) / (2 * a);
}
}

return anzahlNullstellen;
}
```

h) Implement the method:

```
double evaluate(double);
```

```
double Parabel::auswerten(double wert){
return daten[0] * wert * wert + daten[1] * wert + daten[2];
}
```