ECE 39595 C++ Lab, Spring 2021, Test 1

The answer sheet is a separate sheet and can be edited, and therefore annotated with your answers.

Not counting the answer sheet, this exam has 7 pages and 40 questions each worth 2.5 points.

You may begin the exam whenever it becomes available. I will give a 10 minute warning at the end of that the exam answer sheet needs to have already been uploaded to Brightspace.

Answering questions. Most questions are something like `\Q28`. If the statement causes a compile time or runtime error, answer "Err". For the remainder of the program execution, assume the statement that gave an error is not longer part of the program. If the statement has no output and does not cause an error, answer "Ok". If the statement does not cause an error and has output, give the output. If a statement has newlines, you do not need to include those in your answer.

If you are not in zoom with video turned on you may recieve a 0 on the exam. I will be recording the exam.

Programs are given without import statements for brevity. Assume all needed includes are present.

This page, and the following facing page, have questions 1 – 12.

```
class Base {

private:
   virtual void f1( );
public:
   int k = 0;
   Base( );
   virtual void f2(int i);
   virtual void f3( );
   virtual void f4( );
   virtual void f5( );
   void f7( );
};

Base::Base( ) : k(1) { }

void Base::f1( ) {
   std::cout << "B.f1" << std::endl;
}

void Base::f2(int i) {
   std::cout << "B.f2(int)" << std::endl;
}

void Base::f3( ) {
   std::cout << "B.f3" << std::endl;
}

void Base::f4( ) {
   std::cout << "B.f4" << std::endl;
}

void Base::f5( ) {
   std::cout << "B.f5, ";
   f1( );
}

void Base::f7( ) {
   std::cout << "B.f7" << std::endl;
}
```

```
class Derived : public Base {
private:
   virtual void f1( );
public:
   int k;
   Derived( );
   void f2(float f);
   void f3( );
   void f5( );
   virtual void f6( );
   void f7( );
};

Derived::Derived( ) : k(20) { }

void Derived::f1( ) {
   std::cout << "D.f1( )" << std::endl;
}

void Derived::f2(float f) {
   std::cout << "D.f2(f)" << std::endl;
}

void Derived::f3( ) {
   std::cout << "D.f3" << std::endl;
}

void Derived::f5( ) {
   std::cout << "D.f5, ";
   f1( );
}

void Derived::f6( ) {
   std::cout << "D.f6" << std::endl;
}

void Derived::f7( ) {
   std::cout << "D.f7" << std::endl;
}
```

```
int main( ) {
   Base* b = new Base( );
   Derived* d = new Derived;

   b->f1( ); // Q1
   b->f5( ); // Q2
   b->f6( ); // Q3

   b = d;
   b->f2(1.0); // Q4
   b->f3( ); // Q5
   b->f4( ); // Q6
   b->f5( ); // Q7
   b->f6( ); // Q8
   b->f7( ); // Q9
   std::cout << b->k << std::endl; // Q10

   d->f2(1.0); // Q11
   d->f4( ); // Q12
}
```

This page has questions 13 – 17.

```
class B1 {
public:
   B1( );
   ~B1( );
};

B1::B1( ) {
   std::cout << "B1" << std::endl;
}

B1::~B1( ) {
   std::cout << "~B1" << std::endl;
}

class B2 {
public:
   B2( );
   B2(int);
   virtual ~B2( );
};

B2::B2( ) {
   std::cout << "B2" << std::endl;
}

B2::B2(int i) {
   std::cout << "B2(i)" << std::endl;
}

B2::~B2( ) {
   std::cout << "~B2" << std::endl;
}
```

```
class D1 : public B1 {
public:
   D1( );
   virtual ~D1( );
};

D1::D1( ) {
   std::cout << "D1" << std::endl;
}

D1::~D1( ) {
   std::cout << "~D1" << std::endl;
}

class D2 : public B2 {
public:
   D2(int);
   virtual ~D2( );
};

D2::D2(int i) : B2(i) {
   std::cout << "D2(i)" << std::endl;
}

D2::~D2( ) {
   std::cout << "~D2" << std::endl;
}

int main( ) {
   B1* b1 = new D1( ); // Q13
   B2* b2 = new D2(1); // Q14
   D2* d2 = new D2( ); // Q15

   delete b1; // Q16
   delete b2; // Q17
}
```

This page has questions 18 – 25, below.

```
class C1 {
private:
   int i;
public:
   C1(int);
   virtual void f1( ) = 0;
   virtual void f2( ) = 0;
   virtual void f3( );
};

C1::C1(int) {
   i = 10;
   std::cout << "C1" << std::endl;
}

void C1::f3( ) {
   std::cout << "C1.f3" << std::endl;
}

class C2 : public C1 {
public:
   C2(int);
   virtual void f2( );
};

C2::C2(int) : C1(1) {
   std::cout << "C2" << std::endl;
}

void C2::f2( ) {
   std::cout << "C2.f2" << std::endl;
}

class C3 : public C2 {
public:
   C3(int);
   virtual void f1( );
};

C3::C3(int) : C2(1) {
   std::cout << "C3" << std::endl;
}

void C3::f1( ) {
   std::cout << "C3.f1" << std::endl;
}
```

```
class C4 : public C2 {
public:
   C4(int);
   virtual void f1( );
};

C4::C4(int) : C2(1) {
   std::cout << "C4" << std::endl;
}

void C4::f1( ) {
   std::cout << "C4.f1" << std::endl;
}

int main( ) {
   C1* c1 = new C1(1); // Q18
   C2* c2 = new C2(2); // Q19
   C3* c3 = new C3(3); // Q20
   C4* c4 = new C4(4); // Q21

   C1* c = c3;
   c->f1( ); // Q22
   c->f3( ); // Q23

   c = c4;
   c->f1( ); // Q24
   c->f2( ); // Q25
}
```

This page has questions 26 – 35.

```
class B {
public:
   int k = 0;
   B( );
   B(B& oldB);
   virtual void f( );
};

B::B( ) : k(5) { }

B::B(B& oldB) {
   k = 2*oldB.k;
   std::cout << "B&" << std::endl;
}

void B::f( ) {
   std::cout << "B.f" << std::endl;
}

class D : public B {
public:
   int k;
   D( );
   D(D& oldD);
   virtual void f( );
};

D::D( ) : B( ) { };

D::D(D& oldD) : B(oldD) {
   std::cout << "B&" << std::endl;
}

void D::f( ) {std::cout << "D.f" << std::endl;}
```

```
void fooObj(B b) {
   std::cout << "obj B.k: " << b.k << std::endl;
}

void fooRef(B& b) {
   std::cout << "ref B.k: " << b.k << std::endl;
}

void fooPtr(B* b) {
   std::cout << "ptr B.k: " << b->k << std::endl;
   b->k = 500;
   b = nullptr;
}

int main( ) {
   B bObj;
   D dObj;
   B& bR1 = dObj;
   bR1.f( ); // Q26
   bR1 = bObj; // Q27
   bR1.f( ); // Q28

   B& bR2 = dObj;
   fooObj(bObj); // Q29
   fooRef(bObj); // Q30
   fooObj(bR2); // Q31
   fooRef(bR2); // Q32

   B* bPtr = &bObj;
   fooRef(bPtr); // Q33
   fooPtr(bPtr); // Q34
   std::cout << bPtr->k << std::endl; // Q35
}
```

This page intentionally left almost blank

This page, and the following facing page, have questions 1 – 12.

```
class Set {
private:
   int elements[5];
   int size;
public:
   Set( );
   Set(int i);
   Set(Set& s);
   virtual Set& operator+(int _data) const;
   virtual Set& operator-(int _data) const;
   virtual Set& operator-( ) const;
   virtual Set& operator=(Set& s);
   friend std::ostream& operator<<(
      std::ostream& os, const Set& s);
};

Set::Set( ) : size(0) { }

Set::Set(int s) : size(s) {
   for (int i=0; i<size; i++) {
      elements[i] = i;
   }
}

Set::Set(Set& s) {
   size = s.size;
   for (int i=0; i<size; i++) {
      elements[i] = s.elements[i];
   }
}

Set& Set::operator=(Set& s) {
   for (int i=0; i<s.size; i++) {
      elements[i] = s.elements[i];
   }
   size = s.size;
   return -(*this);
}

Set& Set::operator-( ) const {
   Set* s = new Set( );
   s->size = size;
   for (int i=0; i<size; i++) {
      s->elements[i] = -elements[i];
   }
   return *s;
}
```

```
Set& Set::operator-(int _data) const {
   Set* s = new Set( );
   s->size = size;

   int i;
   for (i=0; i<size; i++) {
      if (elements[i] == _data) break;
      s->elements[i] = elements[i];
   }

   for (int j=i; j<size; j++) {
      s->elements[j] = elements[j+1];
   }
   s->size = size-1;
   return *s;
}

Set& Set::operator+(int _data) const {
   Set* s = new Set( );
   for (int i=0; i<size; i++) {
      s->elements[i] = elements[i];
   }

   s->size = size;
   if (s->size == 5) return *s;

   bool hasData = false;

   for (int i=0; i<s->size; i++) {
      if (_data == s->elements[i])
         hasData = true;
   }

   if (!hasData) {
      s->elements[s->size++] = _data;
   }

   return *s;
}

std::ostream& operator<<(
   std::ostream& os, const Set& s) {
   os << "size: " << s.size << ", ";
   for (int i=0; i<s.size; i++) {
      os << s.elements[i] << " ";
   }
   os << std::endl;
   return os;
}
```

```
int main( ) {
   Set s1(3);
   Set s2;
   Set s3;

   std::cout << s1 << std::endl; // Q36

   s1 = -s1;
   std::cout << s1 << std::endl; // Q37

   s1 = s1 - 0;
   std::cout << s1 << std::endl; // Q38

   s1 = s1 + 8;
   std::cout << s1 << std::endl; // Q39

   s2 = s3 = s1;
   std::cout << s2 << std::endl; // Q40
}
```