SOLUTIONS TO MIDTERM

1. **OO concepts** (5 points)

   (a) Working with objects without knowing their precise type; at compile or run time.

   (b) Dynamic binding is necessary for dynamic polymorphism, which is implemented in C++ using virtual functions in inheritance to get overriding done. Static binding achieves static polymorphism: STL and operator overloading are examples in C++. (34 words)

2. **Template and operator overloading** (10 points)

```
template <typename T1, typename T2>
class Tuple {
public:
  Tuple () : ele1(0), ele2(0) {}
  Tuple (const T1& t1, const T2& t2): ele1(t1), ele2(t2) {}
  Tuple operator+ (const Tuple& t) const
    { return Tuple (ele1 + t.ele1, ele2 + t.ele2); }
  T1 get_ele1() const { return ele1; }
  T2 get_ele2() const { return ele2; }
  void set_ele(T1 v1, T2 v2) { ele1 = v1; ele2 = v2; }
private:
  T1 ele1;
  T2 ele2;
};


/*
// if defined outside of the class
template <typename T1, typename T2>
Tuple<T1,T2> Tuple<T1,T2>::operator+ (const Tuple<T1,T2> &t) const
    { return Tuple<T1,T2> (ele1 + t.ele1, ele2 + t.ele2); }
*/


template <typename T1, typename T2>
ostream& operator<< (ostream& os, const Tuple<T1,T2> &t)
{
  os << "(" << t.get_ele1() << ", " << t.get_ele2() << ")" ;
  return os;
}

template <typename T1, typename T2>
Tuple<T1,T2> operator++ (Tuple<T1,T2> &t)
{
   t.set_ele (t.get_ele1()+1, t.get_ele2()+1);
   return t;
}
```

(a) 14 items

(b) 18 items

(a)+(b) = 32 items

10 pts / 32 = 0.3125 pt/item

1

typedef istream_iterator<char> istream_iterator_char;

3. **STL and iterators** (5 points)

```
   // use stl's copy to read characters from cin and copy them to v
1  copy (start, end, dest);
   // use stl's sort to sort v
1  sort(v.begin(), v.end());
   // use stl's to copy the sorted v to cout, each character followed by a space
2  copy (v.begin(), v.end(), ostream_iterator<char> (cout, " "));
```

4. **Order of Construction and Destructio** (5 points)

```
B
1
B
1
R
~R
~1
~B
B
1
R
B
1
L
~L
~1
~B
~R
~1
~B
~1
~B
```

5pt / 22 item = 0.2273 pt/item

5. **Abstract Base Class and Inheritance** (10 points)

```
class BasePoint {
  int x, y; //position
public:
  BasePoint(int px, int py):x(px),y(py) {}
  virtual string type() = 0;
  virtual void info() {
    cout << endl << "figure: " << type() << endl;
    cout << "position: x=" << x << ", y=" << y << endl;
  }
};
class Figure2P : public Figure1P {
  int p2;
public:
  Figure2P(int px, int py, int w, int h):p2(h),Figure1P(px, py, w) {}
  virtual void info() {
    Figure1P::info();
    cout << "property 2: p=" << p2 << endl;
  }
};
```

10 pts / 45 item = 0.222 pt/item