BLG252E – SAMPLE FINAL EXAM ANSWERS

QUESTION 1) [5 points]

```
class Point {
  public:
  float  x,y;
  Point () {};
  Point(float x, float y) : x(x), y(y) {};
};
```

QUESTION 2) [30 points]

```
class LineSegment  {
  public:
  Point P1, P2;
  LineSegment() {};
  LineSegment(Point P1, Point P2) : P1(P1), P2(P2) {};
  Point * find_intersection(const LineSegment &);
};
```

```
Point * LineSegment::find_intersection(const LineSegment & L)   {
   float m1,m2,b1,b2;
   float XInt, YInt; // Intersection point

   if (P2.x == P1.x || L.P2.x == L.P1.x) {
            cout << "Infinite slope found!\n";
            exit(0); // stop program
   }

   m1 = (P2.y - P1.y) / (P2.x - P1.x);
   m2 = (L.P2.y - L.P1.y) / (L.P2.x - L.P1.x);

   if (m1 == m2) // Lines are parallel, so no intersection.
      return NULL;

   b1 = ( (P2.x * P1.y) - (P1.x * P2.y) ) / (P2.x - P1.x);
   b2 = ( (L.P2.x * L.P1.y) - (L.P1.x * L.P2.y) ) / (L.P2.x - L.P1.x);

    // Calculate intersection point coordinates:
      XInt = (b2-b1)/(m1-m2);
      YInt = m1*XInt + b1;

      // A possible intersection found.
      // Now check whether it is within the borders of both line segments.
      if ( ( XInt >= P1.x && XInt <= P2.x || XInt >= P2.x && XInt <= P1.x ) &&
          ( YInt >= P1.y && YInt <= P2.y || YInt >= P2.y && YInt <= P1.y ) &&
          ( XInt >= L.P1.x && XInt <= L.P2.x || XInt >= L.P2.x && XInt <= L.P1.x ) &&
          ( YInt >= L.P1.y && YInt <= L.P2.y || YInt >= L.P2.y && YInt <= L.P1.y ) )
              return new Point(XInt, YInt); // The two line segments intersect.
       else  return NULL; // Not within borders of line segments.

}
```

QUESTION 3) [40 points]

```
class Polygon {
  public:
  float minx,maxx,miny,maxy;
  vector<Point> Pnt;
  Polygon(vector<Point>);
  string status_testing(Polygon &);
};
```

```
Polygon::Polygon(vector<Point> Pnt_in) : Pnt(Pnt_in) {
  int i;
  minx = maxx = Pnt[0].x;
  miny = maxy = Pnt[0].y;
  for (i=0; i<Pnt.size(); i++ )   {
      if (Pnt[i].x < minx)   minx=Pnt[i].x;
      if (Pnt[i].x > maxx)   maxx=Pnt[i].x;
      if (Pnt[i].y < miny)   miny=Pnt[i].y;
      if (Pnt[i].y > maxy)  maxy=Pnt[i].y;
  }
}
```

```
string Polygon::status_testing(Polygon & P)
{
 int i,j,N;
 int ii,jj,M;
 LineSegment L1, L2;
 Point * Pintersection;

 N=Pnt.size();
 M=P.Pnt.size();

 for (i=0; i<=N-1; i++ )
 {
  j=(i+1)%N; // index N-1 becomes index 0
  L1 = LineSegment(Pnt[i], Pnt[j]);

  Pintersection=NULL;

  for (ii=0; ii<=M-1; ii++ )
  {
   jj=(ii+1)%M;
   L2 = LineSegment(P.Pnt[ii], P.Pnt[jj]);
   Pintersection = L1.find_intersection(L2);
   if (Pintersection != NULL)
     return "OVERLAP";
   }//ii

  } //i

 // NO OVERLAP
 if ( (minx >= P.minx && maxx <= P.maxx) &&
     (miny >= P.miny && maxy <= P.maxy) )
    return "CONTAINMENT";    // P is outer

 if ( (P.minx >= minx && P.maxx <= maxx) &&
     (P.miny >= miny && P.maxy <= maxy) )
    return "CONTAINMENT";    // P is inner

 return "DISJOINT";
}
```

QUESTION 4) [10 points]

```
int main()
{
  float x,y;

  Point A1[] = { Point(1,3), Point(2,5), Point(4,6), Point(6,4),
                 Point(5,1), Point(3,4)};
  Polygon Pol1( vector<Point>(A1, A1+5) );

  Point A2[] = {Point(2,1), Point(4,3), Point(7,2.5), Point(6,-2)};
  Polygon Pol2(vector<Point>(A2, A2+3) );

  cout << "Status of Pol1 and Pol2 = ";
  cout << Pol1.status_testing(Pol2);

  return 0;
}
```

QUESTION 5) [15 points]

```
main start

A constructor :10 2 3 ,
B constructor :10 2 3 , 20 3 4 ,
C constructor :10 2 3 , 20 3 4 , 30 4 5 ,

main end

C destructor
B destructor
A destructor
```