Algorithm:

After parsing the output of the A1 function, I use the Make Pairs function on the following pairs in which each pair has their corresponding shapes: AB, CK1, CK2, CK3, CK4, CK5

Note: Although I use A and B as an example, I do this process for ***all*** *the file pairs listed.*

Make Pairs (A, B):

If the length of A is greater than the length of B, we add a shape "DEL"

If the length of B is greater than the length of A, we add a shape "ADD"

Using permutations, return all the combinations of shapes of A and B.

After getting the combinations of the files, I use the Find Matches function.

Find Matches (AB):

For each combination of shapes, I use the descriptions given to see if a shape has changed as in a shape change, if it was added or deleted, if the relationship with other shapes changed, if it moved, or if its size changed.

If the shape has changed in any way, I add the description of that change into a vector and return that.

After finding matches on all the combination of shapes in AB, I compare these matches with CK1, CK2, CK3, CK4, and CK5. I do this in the same way I used Find Matches. I would match a shape pair in AB with a shape pair in CK. Looking at the pairs, I compare the changes that have been made. A change from circle to square in both AB and CK would not result in a penalty. A change from circle to triangle in AB matched with a change from circle to square would. I would also add a penalty for any other changes (See the Penalties for Second Order Transformations section). Using these penalties, I find the lowest penalty. If there are two or more lowest penalties, I move on to calculate the penalties for first order transformations (See the Penalties for First Order Transformations section). This would be the tie breaker for the match. If this does not act as a tie breaker, I will choose the first match that came up.

Penalties for First Order Transformations:

I ranked the transformations from most significant to least significant

Shape Change > Add/Del > Relationships With Other Shapes > Move > Size

I came to this listing because size and move are the simplest transformations with relationships with other shapes as a little more costly as it involves other shapes. Add and delete are very costly but not as costly as a complete shape change. A shape change is like 1 add and 1 delete, but my program won't ever reach a state where there is both an add and a delete as it would just treat it as a shape change. Therefore, the hierarchy is established.

Listing the Transformation and their Costs

Shape Change	10000*(difference in number of sides + 1)
Add/Del	1000
Relationships With Other Shapes	100
Move	10
Size	1

I did this method to ensure that the hierarchy would be intact as a shape change is much more drastic than an add or delete and so on.

In the shape change area, the cost of changing from a four sided polygon to a six sided polygon is greater than the cost of changing from a four sided polygon to a five sided polygon. This is to accommodate for special patterns between two shapes such as the number of side changes. I add 1 to ensure that a shape change is counted even if the number of sides is different (if a square changes to a trapezoid).

If I match a circle in A and a square in B, there would be a penalty of 10000. If I match a circle in A and a circle in B, I add no penalty.

Penalties for Second Order Transformations:

Similar to the first order transformations, I ordered these the same way.

Shape Change > Add/Del > Relationships With Other Shapes > Move > Size

Listing the Transformation and their Costs

Shape Change	10000*(diff in diff num of sides + 1)
Add/Del	1000
Relationships With Other Shapes	100
Move	10
Size	1

I count a shape change penalty when a shape change is in AB and not in CK and vice versa.

The shape change is multiplied by the difference in the difference of the number of sides between AB/CK. This is to accommodate if the transformation from A to B is an increase in one side as well as for CK or any types of these patterns. I add one to ensure that even if there is no difference, there was still a shape change.

I follow this pattern with the other transformations.

If two shape changes are the same, I add no penalty. All the unmatched transformations get a penalty, even if there are no transformations in AB but transformations CK (I would add the penalty of the transformations in CK).

P1

A FILE OUTPUT

```
p1=scc((50.0, 70.0),0,(38.0, 40.0),0,(62.0, 40.0),0,(50.0, 70.0))

p1=scc((50.0, 70.0),0,(38.0, 40.0),0,(62.0, 40.0),0,(50.0, 70.0)) = s4 + s6 + s5

triangle(p1)

area: 360.0

vloc(p1,middle)

hloc(p1,center)

p2=scc((80.0, 30.0),0,(20.0, 30.0),0,(50.0, 90.0),0,(80.0, 30.0))

p2=scc((80.0, 30.0),0,(20.0, 30.0),0,(50.0, 90.0),0,(80.0, 30.0)) = s3 + s1 + s2

triangle(p2)

area: 1800.0

vloc(p2,middle)

hloc(p2,center)

inside(p1,p2)

small(p1)

large(p2)
```

B FILE OUTPUT

```
p1=scc((80.0, 30.0),0,(20.0, 30.0),0,(50.0, 90.0),0,(80.0, 30.0))

p1=scc((80.0, 30.0),0,(20.0, 30.0),0,(50.0, 90.0),0,(80.0, 30.0)) = s3 + s1 + s2

triangle(p1)

area: 1800.0

vloc(p1,middle)
```

```
hloc(p1,center)
```

C FILE OUTPUT

```
c1:[50.0, 50.0, 30.0]
circle(c1)
area: 2827.4333882308138
vloc(c1,middle)
hloc(c1,center)
p1=scc((40.0, 60.0),0,(40.0, 40.0),0,(60.0, 40.0),0,(60.0, 60.0),0,(40.0, 60.0))
p1=scc((40.0, 60.0),0,(40.0, 40.0),0,(60.0, 40.0),0,(60.0, 60.0),0,(40.0, 60.0)) = s4 + s1 + s2 + s3
square(p1)
area: 400.0
vloc(p1,middle)
hloc(p1,center)
inside(p1,c1)
small(p1)
large(c1)
```

K4 FILE OUTPUT

```
c1:[50.0, 50.0, 30.0]
circle(c1)
area: 2827.4333882308138
vloc(c1,middle)
hloc(c1,center)
```

TRANSFORMATION FROM A TO B

```
[('p2', 'p1'), ('p1', 'DEL')]
delete(p1)
```

TRANSFORMATION FROM C TO K

```
[('c1', 'c1'), ('p1', 'DEL')]
delete(p1)
```

TRANSFORMATION FROM AB TO CK

This is in the format:

[[(A1, B1), (C1, K1)], [(A2, B2), (C2, K2)] ...]

```
[[('p2', 'p1'), ('c1', 'c1')], [('p1', 'DEL'), ('p1', 'DEL')]]
```

P2

A FILE OUTPUT

```
c1:[50.0, 75.0, 20.0]
circle(c1)
area: 1256.6370614359173
vloc(c1,top)
hloc(c1,center)
c2:[50.0, 75.0, 10.0]
circle(c2)
area: 314.1592653589793
vloc(c2,top)
hloc(c2,center)
p1=scc((60.0, 15.0),0,(40.0, 15.0),0,(40.0, 35.0),0,(60.0, 35.0),0,(60.0, 15.0))
p1=scc((60.0, 15.0),0,(40.0, 15.0),0,(40.0, 35.0),0,(60.0, 35.0),0,(60.0, 15.0)) = s1 + s4 + s3 + s2
square(p1)
area: 400.0
vloc(p1,bottom)
```

```
hloc(p1,center)
inside(c2,c1)
small(c2)
large(c1)
above(c1,p1)
above(c2,p1)
```

B FILE OUTPUT

```
c1:[50.0, 25.0, 20.0]
circle(c1)
area: 1256.6370614359173
vloc(c1,bottom)
hloc(c1,center)
c2:[50.0, 75.0, 10.0]
circle(c2)
area: 314.1592653589793
vloc(c2,top)
hloc(c2,center)
p1=scc((60.0, 15.0),0,(40.0, 15.0),0,(40.0, 35.0),0,(60.0, 35.0),0,(60.0, 15.0))
p1=scc((60.0, 15.0),0,(40.0, 15.0),0,(40.0, 35.0),0,(60.0, 35.0),0,(60.0, 15.0)) = s1 + s4 + s3 + s2
square(p1)
area: 400.0
vloc(p1,bottom)
hloc(p1,center)
below(c1,c2)
inside(p1,c1)
small(p1)
large(c1)
```

```
above(c2,p1)
```

C FILE OUTPUT

```
c1:[50.0, 25.0, 5.0]
circle(c1)
area: 78.53981633974483
vloc(c1,bottom)
hloc(c1,center)
p1=scc((45.0, 65.0),0,(55.0, 65.0),0,(55.0, 55.0),0,(45.0, 55.0),0,(45.0, 65.0))
p1=scc((45.0, 65.0),0,(55.0, 65.0),0,(55.0, 55.0),0,(45.0, 55.0),0,(45.0, 65.0)) = s4 + s5 + s6 + s7
square(p1)
area: 100.0
vloc(p1,top)
hloc(p1,center)
p2=scc((70.0, 50.0),0,(30.0, 50.0),0,(50.0, 95.0),0,(70.0, 50.0))
p2=scc((70.0, 50.0),0,(30.0, 50.0),0,(50.0, 95.0),0,(70.0, 50.0)) = s1 + s3 + s2
triangle(p2)
area: 900.0
vloc(p2,top)
hloc(p2,center)
below(c1,p1)
below(c1,p2)
below(p1,p2)
inside(p1,p2)
small(p1)
large(p2)
```

K3 FILE OUTPUT

```
c1:[50.0, 25.0, 5.0]

circle(c1)

area: 78.53981633974483

vloc(c1,bottom)

hloc(c1,center)

p1=scc((50.0, 55.0),0,(30.0, 10.0),0,(70.0, 10.0),0,(50.0, 55.0))

p1=scc((50.0, 55.0),0,(30.0, 10.0),0,(70.0, 10.0),0,(50.0, 55.0)) = s3 + s1 + s2

triangle(p1)

area: 900.0

vloc(p1,bottom)

hloc(p1,center)

p2=scc((45.0, 70.0),0,(45.0, 80.0),0,(55.0, 80.0),0,(55.0, 70.0),0,(45.0, 70.0))

p2=scc((45.0, 70.0),0,(45.0, 80.0),0,(55.0, 80.0),0,(55.0, 70.0),0,(45.0, 70.0)) = s7 + s4 + s5 + s6

square(p2)

area: 100.0

vloc(p2,top)

hloc(p2,center)

inside(c1,p1)

small(c1)

large(p1)

below(c1,p2)

below(p1,p2)
```

TRANSFORMATION FROM A TO B

```
[('c1', 'c1'), ('c2', 'c2'), ('p1', 'p1')]

move(top(c1),bottom(c1))

rel(samev(c1,c2),below(c1,c2))

rel(contains(c1,c2),norel(c1,c2))
rel(above(c1,p1),samev(c1,p1))
rel(norel(c1,p1),contains(c1,p1))
```

TRANSFORMATION FROM C TO K

```
[('c1', 'c1'), ('p2', 'p1'), ('p1', 'p2')]
rel(below(c1,p2),samev(c1,p1))
rel(norel(c1,p2),inside(c1,p1))
move(top(p2),bottom(p1))
rel(above(p2,p1),below(p1,p2))
rel(contains(p2,p1),norel(p1,p2))
```

TRANSFORMATION FROM AB TO CK

This is in the format:

[[(A1, B1), (C1, K1)], [(A2, B2), (C2, K2)] ...]

```
[[('p1', 'p1'), ('c1', 'c1')], [('c1', 'c1'), ('p2', 'p1')], [('c2', 'c2'), ('p1', 'p2')]]
change(samev('c1', 'c2') -> below('c1', 'c2'), above('p2', 'p1') -> below('p1', 'p2'))
```

P3

A FILE OUTPUT

```
p1=scc((65.0, 50.0),0,(70.0, 60.0),0,(75.0, 70.0),0,(85.0, 50.0),0,(70.0, 50.0),0,(65.0, 50.0))
p1=scc((65.0, 50.0),0,(75.0, 70.0),0,(85.0, 50.0),0,(65.0, 50.0)) = s10 + s1 + s9 + s8 + s11
triangle(p1)
area: 200.0
vloc(p1,top)
hloc(p1,right)
p2=scc((70.0, 60.0),0,(70.0, 50.0),0,(70.0, 30.0),0,(20.0, 30.0),0,(20.0, 70.0),0,(70.0, 70.0),0,(70.0, 60.0))
```

```
p2=scc((70.0, 30.0),0,(20.0, 30.0),0,(20.0, 70.0),0,(70.0, 70.0),0,(70.0, 30.0)) = s3 + s7 + s6 + s2 + s5 + s4

rectangle(p2)

area: 2000.0

vloc(p2,middle)

hloc(p2,left)

right_of(p1,p2)

above(p1,p2)

overlap(p1,p2)
```

B FILE OUTPUT

```
p1=scc((50.0, 35.0),0,(40.0, 55.0),0,(30.0, 35.0),0,(50.0, 35.0))

p1=scc((50.0, 35.0),0,(40.0, 55.0),0,(30.0, 35.0),0,(50.0, 35.0)) = s3 + s1 + s2

triangle(p1)

area: 200.0

vloc(p1,bottom)

hloc(p1,left)

p2=scc((20.0, 70.0),0,(20.0, 30.0),0,(70.0, 30.0),0,(70.0, 70.0),0,(20.0, 70.0))

p2=scc((20.0, 70.0),0,(20.0, 30.0),0,(70.0, 30.0),0,(70.0, 70.0),0,(20.0, 70.0)) = s5 + s4 + s7 + s6

rectangle(p2)

area: 2000.0

vloc(p2,middle)

hloc(p2,left)

left_of(p1,p2)

below(p1,p2)

inside(p1,p2)

small(p1)

large(p2)
```

C FILE OUTPUT

```
p1=scc((70.0, 60.0),0,(80.0, 60.0),0,(80.0, 70.0),0,(70.0, 70.0),0,(60.0, 70.0),0,(60.0, 60.0),0,(70.0, 60.0))

p1=scc((80.0, 60.0),0,(80.0, 70.0),0,(60.0, 70.0),0,(60.0, 60.0),0,(80.0, 60.0)) = s12 + s11 + s10 + s9 + s8 + s7

rectangle(p1)

area: 200.0

vloc(p1,top)

hloc(p1,right)

p2=scc((5.0, 30.0),0,(10.0, 20.0),0,(70.0, 20.0),0,(70.0, 60.0),0,(70.0, 70.0),0,(70.0, 80.0),0,(10.0, 80.0),0,(5.0, 70.0),0,(5.0, 30.0))

p2=scc((5.0, 30.0),0,(10.0, 20.0),0,(70.0, 20.0),0,(70.0, 80.0),0,(10.0, 80.0),0,(5.0, 70.0),0,(5.0, 30.0)) = s1 + s14 + s13 + s6 + s5 + s4 + s3 + s2

scc(p2)

area: 3850.0

vloc(p2,middle)

hloc(p2,left)

right_of(p1,p2)

above(p1,p2)

overlap(p1,p2)
```

K3 FILE OUTPUT

```
p1=scc((5.0, 30.0),0,(10.0, 20.0),0,(70.0, 20.0),0,(70.0, 80.0),0,(10.0, 80.0),0,(5.0, 70.0),0,(5.0, 30.0))

p1=scc((5.0, 30.0),0,(10.0, 20.0),0,(70.0, 20.0),0,(70.0, 80.0),0,(10.0, 80.0),0,(5.0, 70.0),0,(5.0, 30.0)) = s1 + s6 + s5 + s4 + s3 + s2

scc(p1)

area: 3850.0

vloc(p1,middle)

hloc(p1,left)
```

```
p2=scc((15.0, 40.0),0,(15.0, 30.0),0,(35.0, 30.0),0,(35.0, 40.0),0,(15.0, 40.0))

p2=scc((15.0, 40.0),0,(15.0, 30.0),0,(35.0, 30.0),0,(35.0, 40.0),0,(15.0, 40.0)) = s10 + s9 + s8 + s7

rectangle(p2)

area: 200.0

vloc(p2,bottom)

hloc(p2,left)

right_of(p1,p2)

above(p1,p2)

inside(p2,p1)

small(p2)

large(p1)
```

TRANSFORMATION FROM A TO B

```
[('p1', 'p1'), ('p2', 'p2')]

move(top(p1),bottom(p1))

move(right(p1),left(p1))

rel(right_of(p1,p2),left_of(p1,p2))

rel(above(p1,p2),below(p1,p2))

rel(overlap(p1,p2),inside(p1,p2))
```

TRANSFORMATION FROM C TO K

```
[('p2', 'p1'), ('p1', 'p2')]

rel(left_of(p2,p1),right_of(p1,p2))

rel(below(p2,p1),above(p1,p2))

rel(overlap(p2,p1),contains(p1,p2))

move(top(p1),bottom(p2))

move(right(p1),left(p2))
```

TRANSFORMATION FROM AB TO CK

This is in the format:

[[(A1, B1), (C1, K1)], [(A2, B2), (C2, K2)] ...]

```
[[('p2', 'p2'), ('p2', 'p1')], [('p1', 'p1'), ('p1', 'p2')]]
```

P4

A FILE OUTPUT

```
c1:[50.0, 50.0, 10.0]
circle(c1)
area: 314.1592653589793
vloc(c1,middle)
hloc(c1,center)
p1=scc((80.0, 30.0),0,(20.0, 30.0),0,(50.0, 90.0),0,(80.0, 30.0))
p1=scc((80.0, 30.0),0,(20.0, 30.0),0,(50.0, 90.0),0,(80.0, 30.0)) = s3 + s1 + s2
triangle(p1)
area: 1800.0
vloc(p1,middle)
hloc(p1,center)
inside(c1,p1)
small(c1)
large(p1)
```

B FILE OUTPUT

```
c1:[50.0, 75.0, 20.0]
circle(c1)
area: 1256.6370614359173
vloc(c1,top)
```

```
hloc(c1,center)
p1=scc((38.0, 20.0),0,(50.0, 50.0),0,(62.0, 20.0),0,(38.0, 20.0))
p1=scc((38.0, 20.0),0,(50.0, 50.0),0,(62.0, 20.0),0,(38.0, 20.0)) = s1 + s2 + s3
triangle(p1)
area: 360.0
vloc(p1,bottom)
hloc(p1,center)
above(c1,p1)
```

C FILE OUTPUT

```
c1:[50.0, 50.0, 20.0]
circle(c1)
area: 1256.6370614359173
vloc(c1,middle)
hloc(c1,center)
p1=scc((60.0, 60.0),0,(40.0, 60.0),0,(40.0, 40.0),0,(60.0, 40.0),0,(60.0, 60.0))
p1=scc((60.0, 60.0),0,(40.0, 60.0),0,(40.0, 40.0),0,(60.0, 40.0),0,(60.0, 60.0)) = s1 + s4 + s3 + s2
square(p1)
area: 400.0
vloc(p1,middle)
hloc(p1,center)
inside(p1,c1)
small(p1)
large(c1)
```

K2 FILE OUTPUT

```
c1:[50.0, 20.0, 10.0]
circle(c1)
```

```
area: 314.1592653589793
vloc(c1,bottom)
hloc(c1,center)
p1=scc((30.0, 90.0),0,(70.0, 90.0),0,(70.0, 50.0),0,(30.0, 50.0),0,(30.0, 90.0))
p1=scc((30.0, 90.0),0,(70.0, 90.0),0,(70.0, 50.0),0,(30.0, 50.0),0,(30.0, 90.0)) = s1 + s2 + s3 + s4
square(p1)
area: 1600.0
vloc(p1,top)
hloc(p1,center)
below(c1,p1)
```

TRANSFORMATION FROM A TO B

```
[('c1', 'c1'), ('p1', 'p1')]
change(small(c1),large(c1))
move(middle(c1),top(c1))
rel(samev(c1,p1),above(c1,p1))
rel(inside(c1,p1),norel(c1,p1))
change(large(p1),small(p1))
move(middle(p1),bottom(p1))
```

TRANSFORMATION FROM C TO K

```
[('c1', 'c1'), ('p1', 'p1')]
change(large(c1),small(c1))
move(middle(c1),bottom(c1))
rel(samev(c1,p1),below(c1,p1))
rel(contains(c1,p1),norel(c1,p1))
change(small(p1),large(p1))
move(middle(p1),top(p1))
```

TRANSFORMATION FROM AB TO CK

This is in the format:

[[(A1, B1), (C1, K1)], [(A2, B2), (C2, K2)] ...]

```
[[('p1', 'p1'), ('c1', 'c1')], [('c1', 'c1'), ('p1', 'p1')]]
```

P5

A FILE OUTPUT

```
d1:[50.0, 80.0]
area: 0
dot(d1)
vloc(d1,top)
hloc(d1,center)
p1=scc((40.0, 40.0),0,(60.0, 40.0),0,(60.0, 20.0),0,(40.0, 20.0),0,(40.0, 40.0))
p1=scc((40.0, 40.0),0,(60.0, 40.0),0,(60.0, 20.0),0,(40.0, 20.0),0,(40.0, 40.0)) = s1 + s2 + s3 + s4
square(p1)
area: 400.0
vloc(p1,bottom)
hloc(p1,center)
above(d1,p1)
```

B FILE OUTPUT

```
p1=scc((40.0, 40.0),0,(60.0, 40.0),0,(60.0, 20.0),0,(40.0, 20.0),0,(40.0, 40.0))
p1=scc((40.0, 40.0),0,(60.0, 40.0),0,(60.0, 20.0),0,(40.0, 20.0),0,(40.0, 40.0)) = s1 + s2 + s3 + s4
square(p1)
area: 400.0
vloc(p1,bottom)
```

```
hloc(p1,center)
```

C FILE OUTPUT

```
d1:[30.0, 50.0]
area: 0
dot(d1)
vloc(d1,middle)
hloc(d1,left)
p1=scc((80.0, 40.0),0,(80.0, 60.0),0,(60.0, 60.0),0,(60.0, 40.0),0,(80.0, 40.0))
p1=scc((80.0, 40.0),0,(80.0, 60.0),0,(60.0, 60.0),0,(60.0, 40.0),0,(80.0, 40.0)) = s2 + s1 + s4 + s3
square(p1)
area: 400.0
vloc(p1,middle)
hloc(p1,right)
left_of(d1,p1)
```

K5 FILE OUTPUT

```
p1=scc((80.0, 40.0),0,(80.0, 60.0),0,(60.0, 60.0),0,(60.0, 40.0),0,(80.0, 40.0))
p1=scc((80.0, 40.0),0,(80.0, 60.0),0,(60.0, 60.0),0,(60.0, 40.0),0,(80.0, 40.0)) = s2 + s1 + s4 + s3
square(p1)
area: 400.0
vloc(p1,middle)
hloc(p1,right)
```

TRANSFORMATION FROM A TO B

```
[('p1', 'p1'), ('d1', 'DEL')]
delete(d1)
```

TRANSFORMATION FROM C TO K

[('p1', 'p1'), ('d1', 'DEL')]

delete(d1)

TRANSFORMATION FROM AB TO CK

This is in the format:

[[(A1, B1), (C1, K1)], [(A2, B2), (C2, K2)] ...]

[[('p1', 'p1'), ('p1', 'p1')], [('d1', 'DEL'), ('d1', 'DEL')]]