1) A Beam transform such as Pardo modifies the input collection while processing its elements.

A. True  
B. False
2) What kind of object does the ParDo transform expect?

A. A DoFn subclass
B. A DoFn super class
C. A DoFn abstract class
3) Does ParDo support random access to PCollections? For example, is the highlighted code allowed?

A. Yes
B. No

class ComputeWordLengthFn(beam.DoFn):
    def process(self, element):
        another_element = words[3]
        if len(element) > len(another_element):
            return [len(element)]

word_lengths = words | beam.ParDo(ComputeWordLengthFn())
4) Which Beam transform is equivalent to a SQL \texttt{WHERE} clause?

A. ParDo
B. GroupByKey
C. CoGroupByKey
D. Flatten
5) Which Beam transform is equivalent to a SQL JOIN?

A. ParDo
B. GroupByKey
C. CoGroupByKey
D. Flatten
ParDo Transform

- Maps 1 input element to (1, 0, many) output elements
- Invokes a user-specified function on each of the elements of the input PCollection
- User code is implemented as a subclass of DoFn with a process(self, element) method
- Input elements are processed independently and in parallel
- Output elements are bundled into a new PCollection
- Typical usage: filtering, formatting, extracting parts of data, performing computations on data elements
GroupByKey Transform

- Takes a PCollection as input where each element is a (key, value) pair
- Groups the values by unique key
- Produces a PCollection as output where each element is a (key, list(value)) pair
- Related, but not analogous to `GROUP BY` in SQL
Demo: Student_single.py

git clone https://github.com/cs327e-spring2019/snippets.git
Hands-on Exercise 1

Run Student_single.py
iClicker Question 1

How many records are in the resulting Student table?

A. 0
B. 12
C. 15
Demo: convert pipeline to Dataflow

git clone https://github.com/cs327e-spring2019/snippets.git

Walk through Student_cluster.py
Hands-on Exercise 2

Create `Teacher_cluster.py` from `Teacher_single.py`

Run `Teacher_cluster.py` on Dataflow
iClicker Question 2

How many nodes are in the job’s execution graph?

A. 3  
B. 4  
C. 9
ParDo Side Inputs

- A side input is an optional input passed to DoFn
- Passed as extra arguments to `process(self, element, side_input1, side_input2 ...)`
- Side input can be ordinary values or entire PCollections
- DoFn reads side input while processing an individual element
- Multiple side inputs per DoFn are supported
Demo: Side input example

git clone https://github.com/cs327e-spring2019/snippets.git

Walk through Takes_single.py
Flatten Transform

- Takes a list of PCollections as input
- Produces a single PCollection as output
- Results contain all the elements from the input PCollections
- Note: Input PCollections must have matching schemas

```python
a_pcoll = p | 'Read File 1' >> ReadFromText('oscars_data_archive.tsv')
b_pcoll = p | 'Read File 2' >> ReadFromText('oscars_data_2019.tsv')

# Union the two PCollections
c_pcoll = (a_pcoll, b_pcoll) | 'Merge PCollections' >> beam.Flatten()
```
CoGroupByKey Transform

- Takes two or more PCollections as input
- Every element in the input is a (key, value) pair
- Groups values from all input PCollections by common key
- Produces a PCollection as output where each element is a (key, value) pair
- Output value is a list of dictionaries containing all data associated with unique key
- Analogous to the FULL OUTER JOIN in SQL
CoGroupByKey Transform

```plaintext
q1 = 'SELECT sid, cno, grade FROM college_split.Takes'
q2 = 'SELECT cno, cname FROM college_split.Class'

takes_pcoll = p | 'Run Q1' >> beam.io.Read(beam.io.BigQuerySource(query=q1))
class_pcoll = p | 'Run Q2' >> beam.io.Read(beam.io.BigQuerySource(query=q2))

takes_tuple = takes_pcoll | 'Takes Tuple' >> beam.ParDo(MakeTuple())
class_tuple = class_pcoll | 'Class Tuple' >> beam.ParDo(MakeTuple())

joined_pcoll = (takes_tuple, class_tuple) | 'Join' >> beam.CoGroupByKey()
```
Milestone 6

1) Requirements and rubric: assignment sheet

2) Debugging assistance: sign-up sheet