(3)(i) Run a basic word Count Map Reduce program to Understand Map reduce paradigm.

solution:1

->

```
from Collections import defaultdict
import multiprocessing

# Mapper function
def Mapper(chunk):
    word_count = defaultdict(int)
    for word in chunk.split()
        word_count[word] += 1
    return word_count

# Reducer function

def reducer(results):
    word_count = defaultdict(int)
    for wc in result:
        for word, count in wc.items():
            word_count[word] += count
    return word_count

# MapperReducer function

def mapperreduce(data, mapper, reducer, num_process=2)
    with multiprocessing.pool(processes = num_process) as pool:
        mapped_data = pool.map(mapper, data)
        reduced_data = reducer(mapped_data)
    return reduced_data.

if __name__ == "__main__":
```

```
# Sample input data
data = [
    "hello world",
    "Word is beautiful",
    "hello beautiful world"
]

# Execute MapReduce
word_counts = mapreduce(data, Mapper, reducer)

# Output results
print("word counts:");
for word, count in word_counts.items():
    printf"{word}: {count}")
```

Solution-2

```
from collections import defaultdict
from multiprocessing import pool

# Mapper function

    def mapper(text):
        word_count = defaultdict(int)
    for word in text.split():
        word_count[word.lower()] += 1
    return word_count

# Reducer function

    def reducer(count_dicts):
        word_count = defaultdict(int)
    for word, count in count_dict.item():
        word_count[word] += count
    return word_count

# Main function to orchestrate MapReduce process
    def main():
        # Sample input data
        data = [
                "Hello world",
                "world is beautiful",
                "Hello beautiful world"
            ]
# Map phase
        with pool() as pool:
            Mapped_data = pool.map(mapper, data)
```

```
# Reduce phase
reduced_result = reducer(mapped_data)

# Print the final word counts
printf("Word Count:")

for word, count in reduced_result.items():
    print(f"{word}: {count}")

if __name__ == "__main__":
    main()
```

(3) (iii) Mapreduce program in python using external data set with excel file:

```
-1

import pandas as pd
import multiprocessing
from collection import defultdict

def mapper(data):
    word_count = defultdict(int)
    for row in data.iertuples():

        for word in str(row.text).split():
            word_count[word]+=1
        return word_count

def reducer(word_counts):
    final_word_count = defaultdict(int)
        for word_count in word_count():
            final_word_count[word]+=count
    return final_word_count

def mapreducer(data, num_workers=2):
    pool = multiprocessing.pool(process=num_workers)
    chunk = [data[i:i+chunk_size] for i in range
                                    (0, len(data), chunk_size]
    word_count = pool.map(mapper, chunks)
    final_word_count = reducer(word_counts)
    pool.close()
     pool.join()
     return final_word_count
if __name__ == "__main__":
```

```
# Read data from Excel file
df = pd.read_excel('your_excel_file.xlsx')

Excel file
    result = mupersdue(df, num_workers=2)
    print(result)
```