MapReduce Simulation: Word Count Program in C++

Submitted by:

Aqsa Fayaz 22i1865

Muntaha Asim 22i1868

Manal Amir 22i1940

Introduction:

This program is a simulation of a MapReduce framework implemented in C++ using **processes**, **pipes**, **threads**, and **semaphores**. The program processes textual input to count the occurrences of words. It is divided into three main phases:

1. **Map Phase**: Words are distributed among multiple mappers, which process them to produce intermediate (key, value) pairs.
2. **Shuffle Phase**: The intermediate results are sorted and grouped by keys.
3. **Reduce Phase**: Reducer threads aggregate the values for each key to produce the final word count.

Program Design:

1. Input Processing

- The program begins by reading user input (multi-line text).
- Punctuation is removed using remove_punctuation().
- The text is split into words using split_into_words(), which tokenizes the input into an array of strings (words).

2. Dynamic Mapper Allocation

- The number of mappers is dynamically calculated based on the number of words (e.g., one mapper per 20 words).
- **Pipes** are used to facilitate communication between the parent process (distributor) and child processes (mappers).

3. Mapping Phase

- Each mapper processes a chunk of words. The parent sends words to each mapper via a write-end pipe, and each mapper writes (key, value) pairs to the parent through the output pipe.
- The parent collects intermediate results from all mappers.

4. Shuffle Phase

- The intermediate results are sorted by key using insertion_sort(). This ensures that all occurrences of the same word are grouped together for efficient reduction.

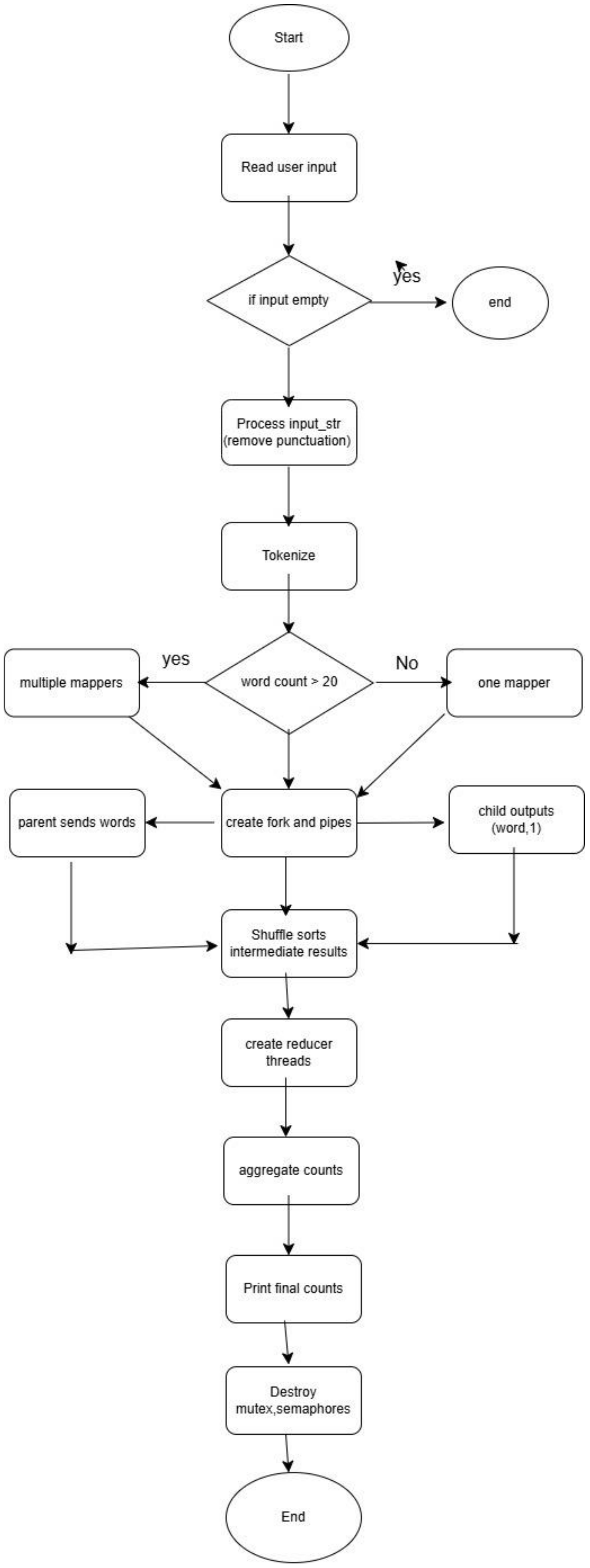
5. Reducing Phase

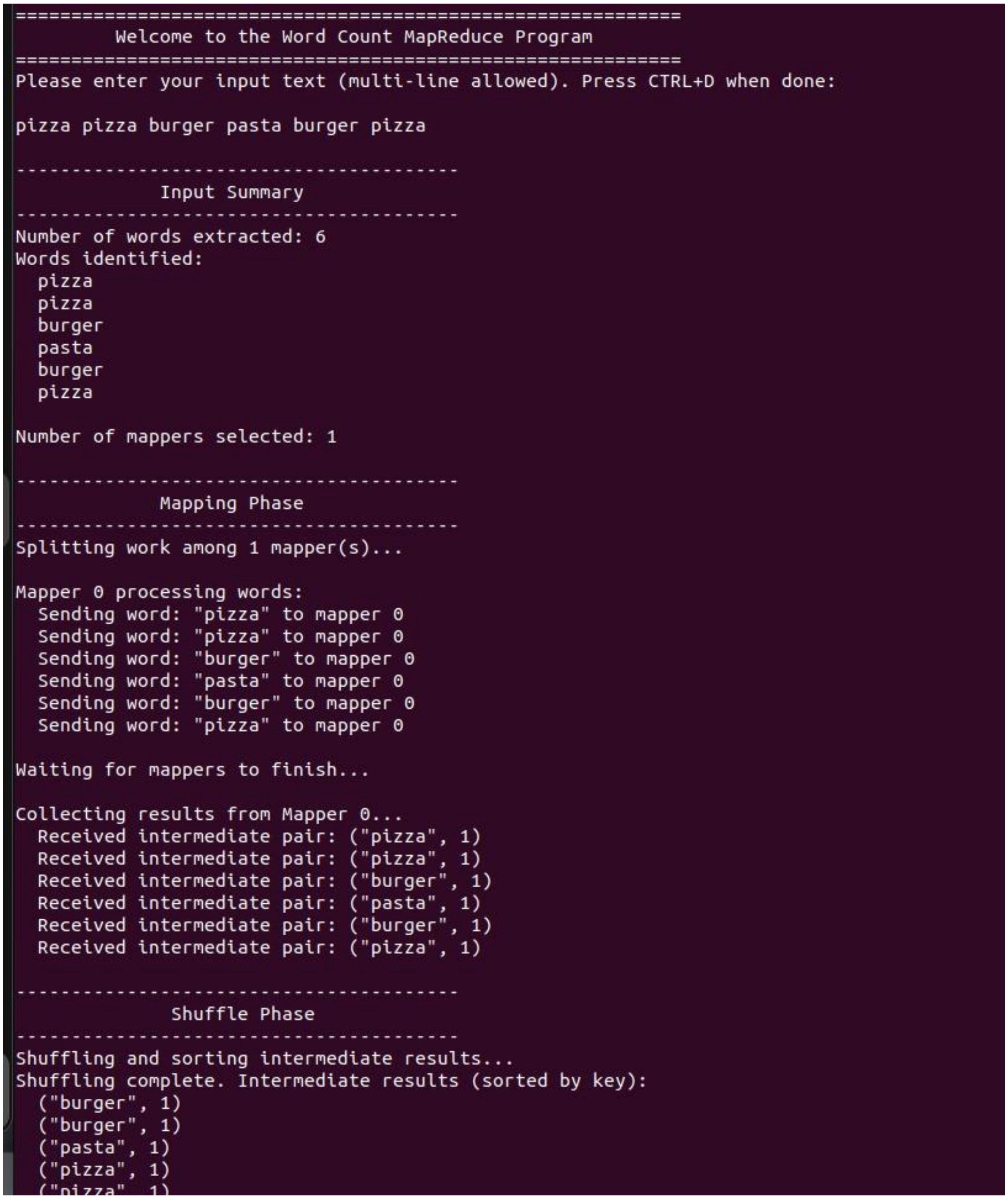
- Reducer threads are created for each unique key.
- Each reducer thread processes a subset of intermediate results corresponding to a single key and aggregates the values.
- The final word counts are printed to the console.

Code Walkthrough:

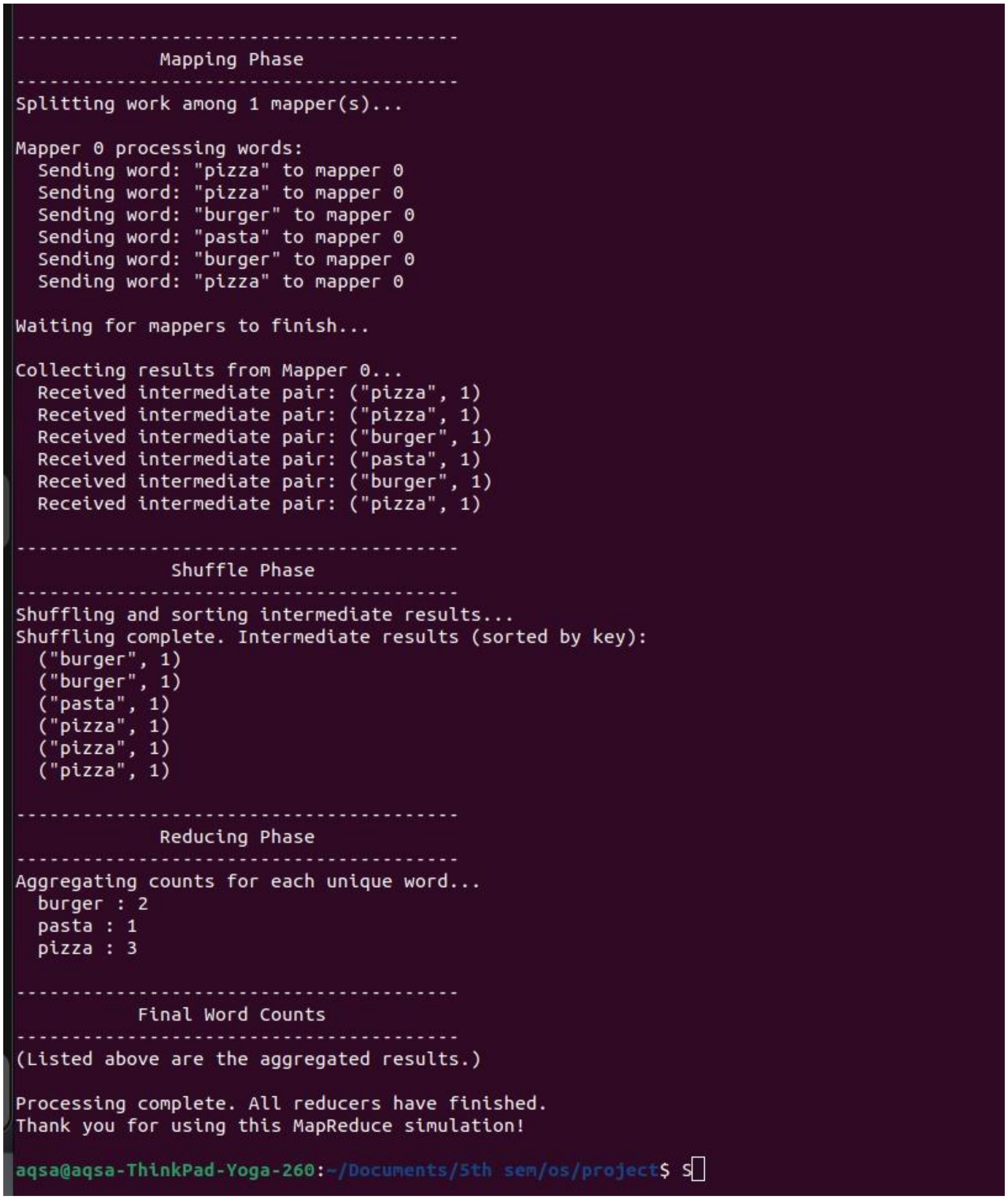
Key Functions

1. **remove_punctuation()**
   - Removes all punctuation characters from the input string.
   - Example: "hello, world!" becomes "hello world".
2. **split_into_words()**
   - Tokenizes the input string into individual words stored in a 2D array.
   - Example: "hello world" → ["hello", "world"].
3. **insert_intermediate_pair()**
   - Inserts a (key, value) pair into the shared intermediate data structure.
   - Thread-safe using a mutex.
4. **insertion_sort()**
   - Sorts the intermediate results by key using the insertion sort algorithm.
5. **mapper_process()**
   - Child process function for mappers. Reads words from a pipe, processes them into (key, value) pairs, and writes them to another pipe.
6. **reducer_thread()**
   - Thread function for reducers. Aggregates values for a specific key and prints the result.

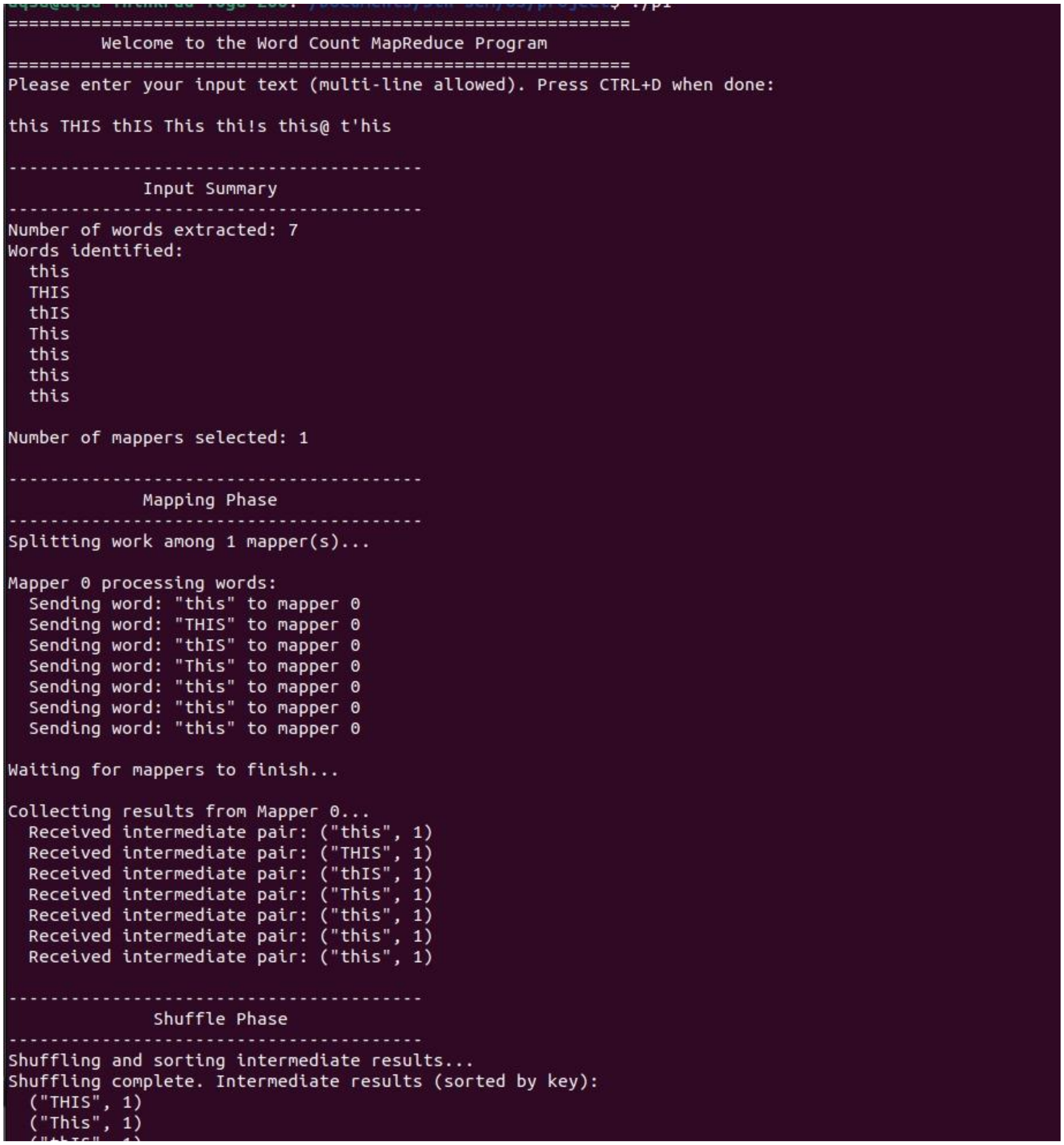

Start
Read user input
if input empty
yes
end
Process input_str (remove punctuation)
Tokenize
word count > 20
yes
No
multiple mappers
one mapper
create fork and pipes
parent sends words
child outputs (word,1)
Shuffle sorts intermediate results
create reducer threads
aggregate counts
Print final counts
Destroy mutex,semaphores
End



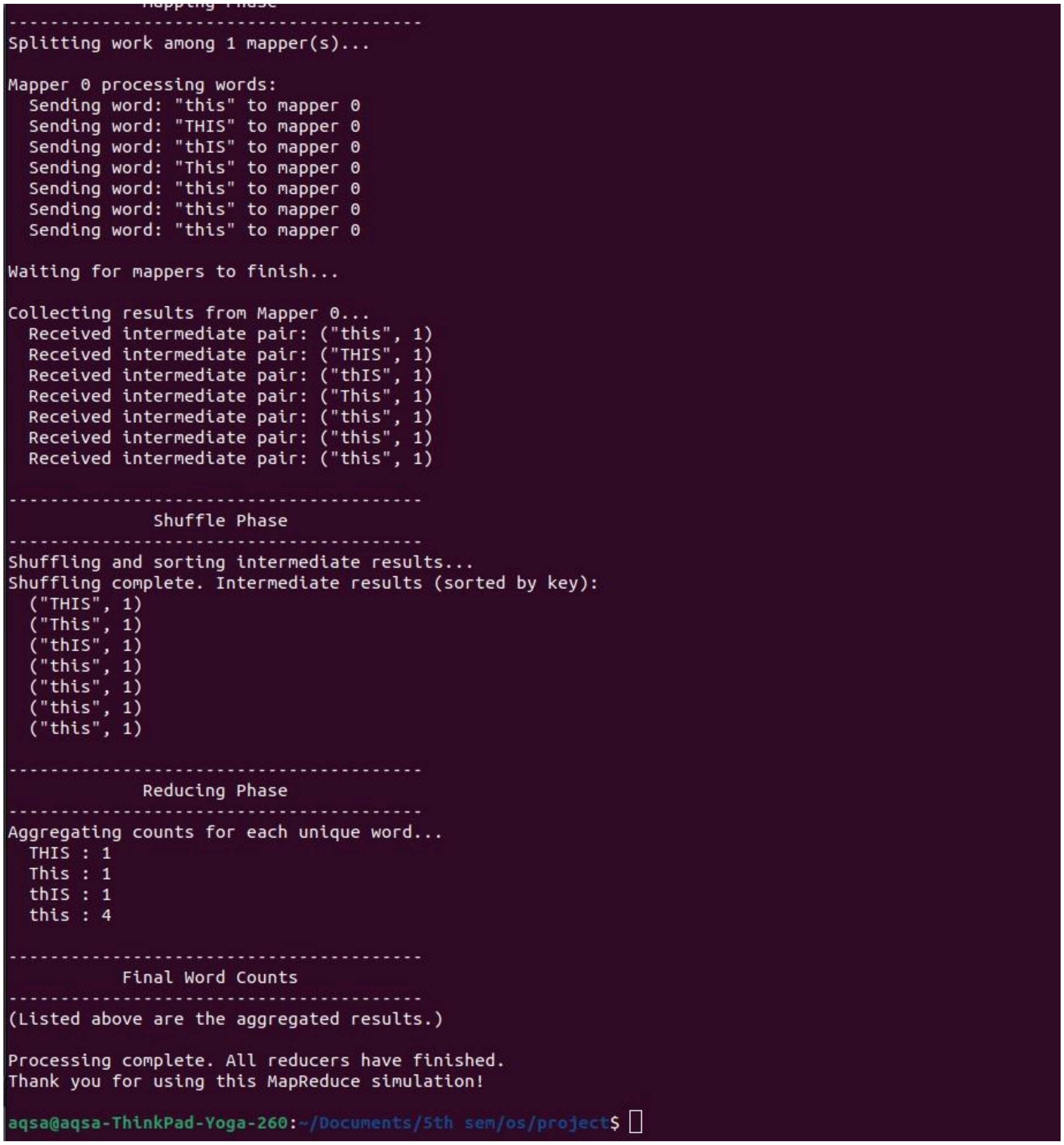
```
============================================================
         Welcome to the Word Count MapReduce Program
============================================================
Please enter your input text (multi-line allowed). Press CTRL+D when done:

pizza pizza burger pasta burger pizza

----------------------------------------
             Input Summary
----------------------------------------
Number of words extracted: 6
Words identified:
  pizza
  pizza
  burger
  pasta
  burger
  pizza

Number of mappers selected: 1

----------------------------------------
             Mapping Phase
----------------------------------------
Splitting work among 1 mapper(s)...

Mapper 0 processing words:
  Sending word: "pizza" to mapper 0
  Sending word: "pizza" to mapper 0
  Sending word: "burger" to mapper 0
  Sending word: "pasta" to mapper 0
  Sending word: "burger" to mapper 0
  Sending word: "pizza" to mapper 0

Waiting for mappers to finish...

Collecting results from Mapper 0...
  Received intermediate pair: ("pizza", 1)
  Received intermediate pair: ("pizza", 1)
  Received intermediate pair: ("burger", 1)
  Received intermediate pair: ("pasta", 1)
  Received intermediate pair: ("burger", 1)
  Received intermediate pair: ("pizza", 1)

----------------------------------------
              Shuffle Phase
----------------------------------------
Shuffling and sorting intermediate results...
Shuffling complete. Intermediate results (sorted by key):
  ("burger", 1)
  ("burger", 1)
  ("pasta", 1)
  ("pizza", 1)
  ("pizza", 1)
```



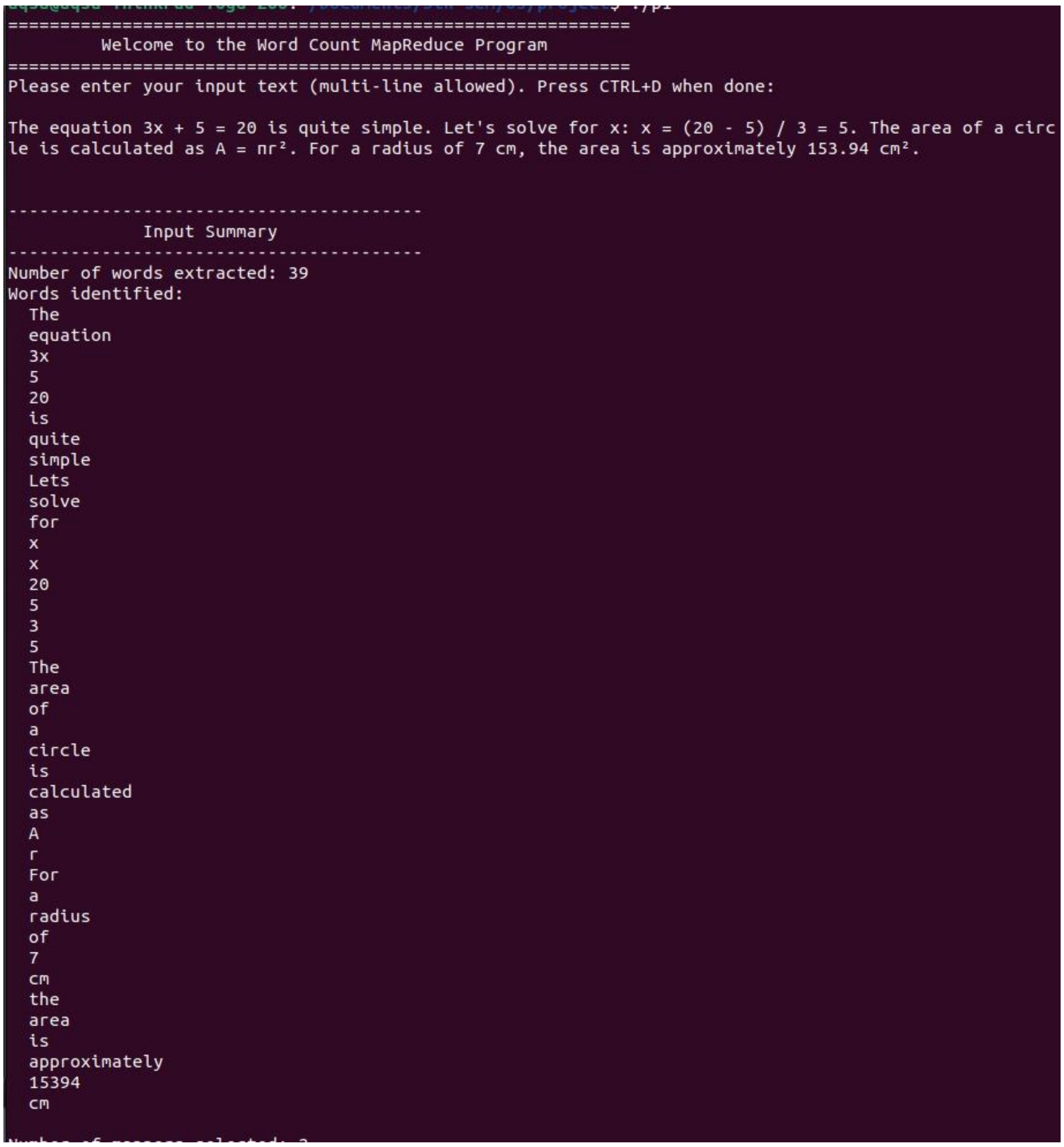
```
----------------------------------------
             Mapping Phase
----------------------------------------
Splitting work among 1 mapper(s)...

Mapper 0 processing words:
  Sending word: "pizza" to mapper 0
  Sending word: "pizza" to mapper 0
  Sending word: "burger" to mapper 0
  Sending word: "pasta" to mapper 0
  Sending word: "burger" to mapper 0
  Sending word: "pizza" to mapper 0

Waiting for mappers to finish...

Collecting results from Mapper 0...
  Received intermediate pair: ("pizza", 1)
  Received intermediate pair: ("pizza", 1)
  Received intermediate pair: ("burger", 1)
  Received intermediate pair: ("pasta", 1)
  Received intermediate pair: ("burger", 1)
  Received intermediate pair: ("pizza", 1)

----------------------------------------
              Shuffle Phase
----------------------------------------
Shuffling and sorting intermediate results...
Shuffling complete. Intermediate results (sorted by key):
  ("burger", 1)
  ("burger", 1)
  ("pasta", 1)
  ("pizza", 1)
  ("pizza", 1)
  ("pizza", 1)

----------------------------------------
             Reducing Phase
----------------------------------------
Aggregating counts for each unique word...
  burger : 2
  pasta : 1
  pizza : 3

----------------------------------------
           Final Word Counts
----------------------------------------
(Listed above are the aggregated results.)

Processing complete. All reducers have finished.
Thank you for using this MapReduce simulation!

aqsa@aqsa-ThinkPad-Yoga-260:~/Documents/5th sem/os/project$ S
```



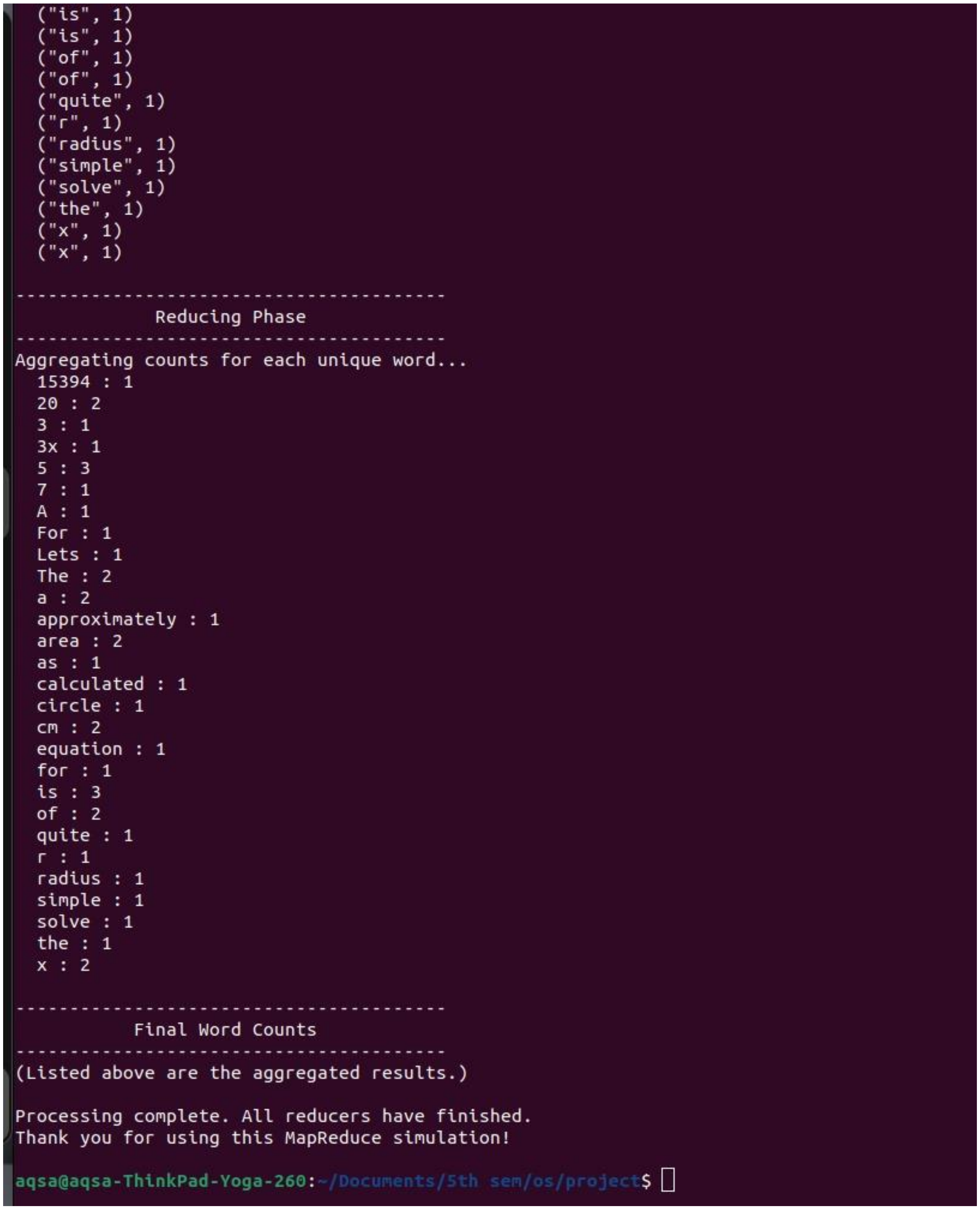
```
============================================================
         Welcome to the Word Count MapReduce Program
============================================================
Please enter your input text (multi-line allowed). Press CTRL+D when done:

this THIS thIS This thi!s this@ t'his

----------------------------------------
             Input Summary
----------------------------------------
Number of words extracted: 7
Words identified:
  this
  THIS
  thIS
  This
  this
  this
  this

Number of mappers selected: 1

----------------------------------------
             Mapping Phase
----------------------------------------
Splitting work among 1 mapper(s)...

Mapper 0 processing words:
  Sending word: "this" to mapper 0
  Sending word: "THIS" to mapper 0
  Sending word: "thIS" to mapper 0
  Sending word: "This" to mapper 0
  Sending word: "this" to mapper 0
  Sending word: "this" to mapper 0
  Sending word: "this" to mapper 0

Waiting for mappers to finish...

Collecting results from Mapper 0...
  Received intermediate pair: ("this", 1)
  Received intermediate pair: ("THIS", 1)
  Received intermediate pair: ("thIS", 1)
  Received intermediate pair: ("This", 1)
  Received intermediate pair: ("this", 1)
  Received intermediate pair: ("this", 1)
  Received intermediate pair: ("this", 1)

----------------------------------------
              Shuffle Phase
----------------------------------------
Shuffling and sorting intermediate results...
Shuffling complete. Intermediate results (sorted by key):
  ("THIS", 1)
  ("This", 1)
```



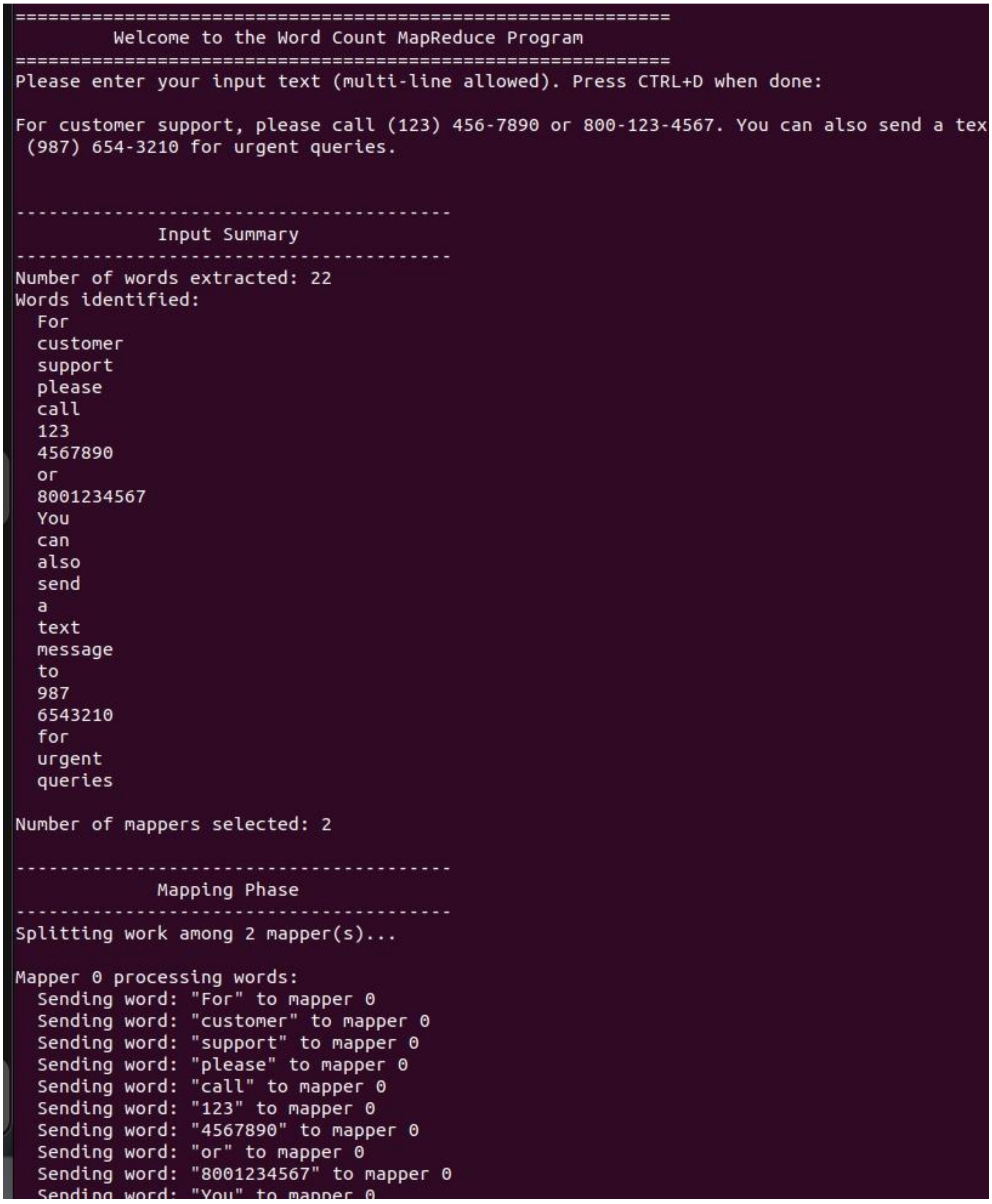
```
----------------------------------------
Splitting work among 1 mapper(s)...

Mapper 0 processing words:
  Sending word: "this" to mapper 0
  Sending word: "THIS" to mapper 0
  Sending word: "thIS" to mapper 0
  Sending word: "This" to mapper 0
  Sending word: "this" to mapper 0
  Sending word: "this" to mapper 0
  Sending word: "this" to mapper 0

Waiting for mappers to finish...

Collecting results from Mapper 0...
  Received intermediate pair: ("this", 1)
  Received intermediate pair: ("THIS", 1)
  Received intermediate pair: ("thIS", 1)
  Received intermediate pair: ("This", 1)
  Received intermediate pair: ("this", 1)
  Received intermediate pair: ("this", 1)
  Received intermediate pair: ("this", 1)

----------------------------------------
              Shuffle Phase
----------------------------------------
Shuffling and sorting intermediate results...
Shuffling complete. Intermediate results (sorted by key):
  ("THIS", 1)
  ("This", 1)
  ("thIS", 1)
  ("this", 1)
  ("this", 1)
  ("this", 1)
  ("this", 1)

----------------------------------------
             Reducing Phase
----------------------------------------
Aggregating counts for each unique word...
  THIS : 1
  This : 1
  thIS : 1
  this : 4

----------------------------------------
           Final Word Counts
----------------------------------------
(Listed above are the aggregated results.)

Processing complete. All reducers have finished.
Thank you for using this MapReduce simulation!

aqsa@aqsa-ThinkPad-Yoga-260:~/Documents/5th sem/os/project$ 
```



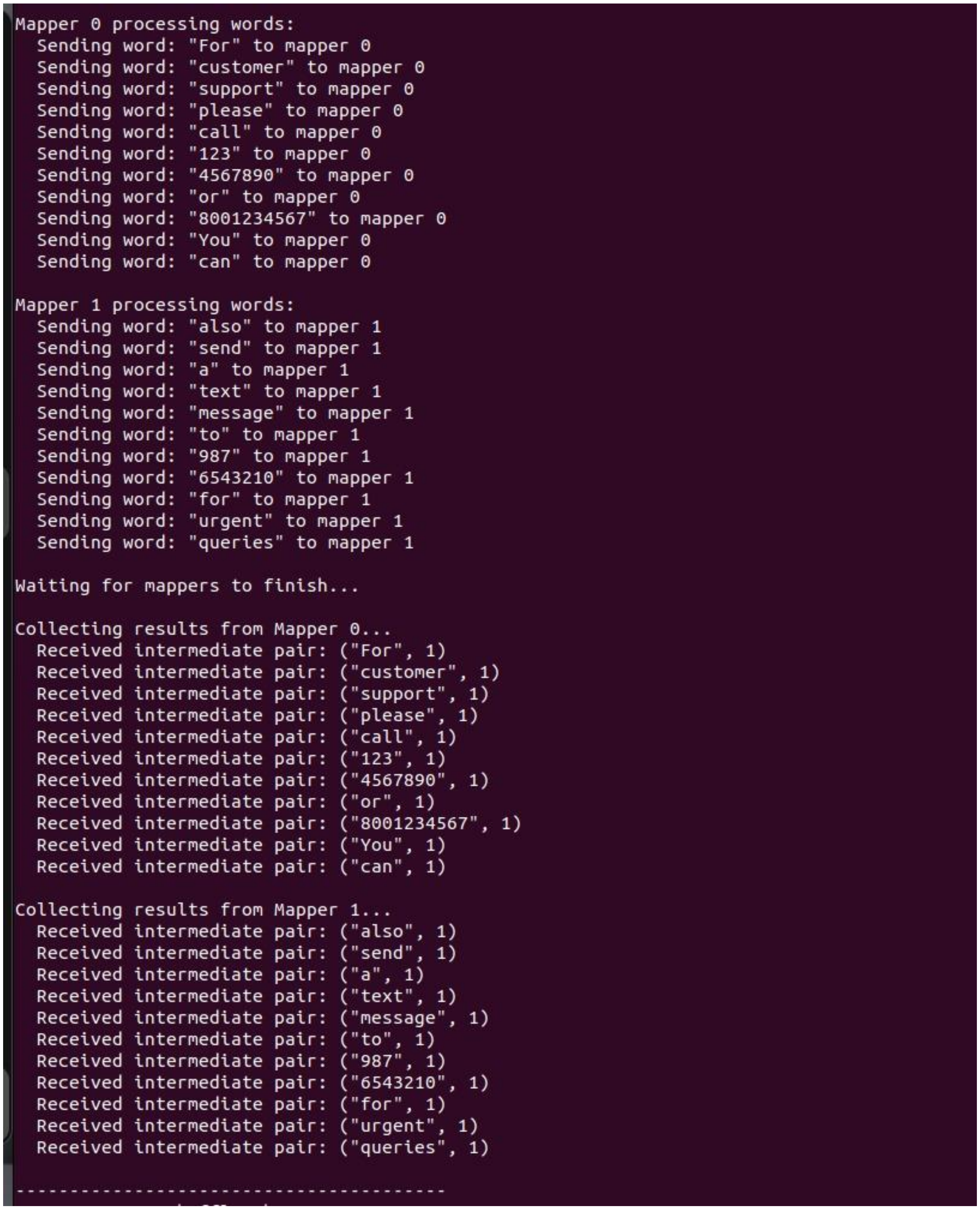
```
============================================================
         Welcome to the Word Count MapReduce Program
============================================================
Please enter your input text (multi-line allowed). Press CTRL+D when done:

The equation 3x + 5 = 20 is quite simple. Let's solve for x: x = (20 - 5) / 3 = 5. The area of a circ
le is calculated as A = πr². For a radius of 7 cm, the area is approximately 153.94 cm².


----------------------------------------
             Input Summary
----------------------------------------
Number of words extracted: 39
Words identified:
  The
  equation
  3x
  5
  20
  is
  quite
  simple
  Lets
  solve
  for
  x
  x
  20
  5
  3
  5
  The
  area
  of
  a
  circle
  is
  calculated
  as
  A
  r
  For
  a
  radius
  of
  7
  cm
  the
  area
  is
  approximately
  15394
  cm
```



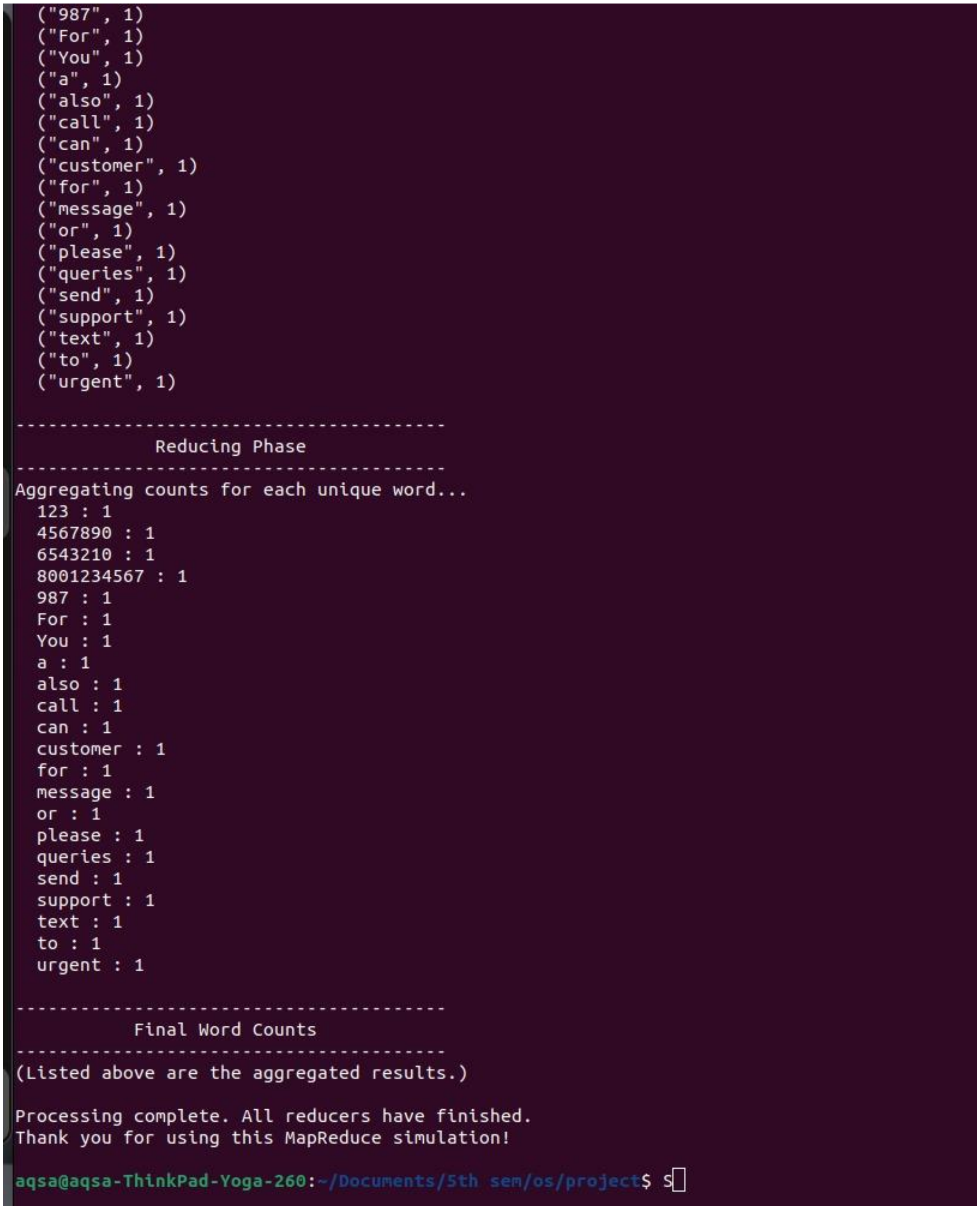
```
  ("is", 1)
  ("is", 1)
  ("of", 1)
  ("of", 1)
  ("quite", 1)
  ("r", 1)
  ("radius", 1)
  ("simple", 1)
  ("solve", 1)
  ("the", 1)
  ("x", 1)
  ("x", 1)

----------------------------------------
             Reducing Phase
----------------------------------------
Aggregating counts for each unique word...
  15394 : 1
  20 : 2
  3 : 1
  3x : 1
  5 : 3
  7 : 1
  A : 1
  For : 1
  Lets : 1
  The : 2
  a : 2
  approximately : 1
  area : 2
  as : 1
  calculated : 1
  circle : 1
  cm : 2
  equation : 1
  for : 1
  is : 3
  of : 2
  quite : 1
  r : 1
  radius : 1
  simple : 1
  solve : 1
  the : 1
  x : 2

----------------------------------------
           Final Word Counts
----------------------------------------
(Listed above are the aggregated results.)

Processing complete. All reducers have finished.
Thank you for using this MapReduce simulation!

aqsa@aqsa-ThinkPad-Yoga-260:~/Documents/5th sem/os/project$ 
```



```
============================================================
         Welcome to the Word Count MapReduce Program
============================================================
Please enter your input text (multi-line allowed). Press CTRL+D when done:

For customer support, please call (123) 456-7890 or 800-123-4567. You can also send a tex
 (987) 654-3210 for urgent queries.



----------------------------------------
             Input Summary
----------------------------------------
Number of words extracted: 22
Words identified:
  For
  customer
  support
  please
  call
  123
  4567890
  or
  8001234567
  You
  can
  also
  send
  a
  text
  message
  to
  987
  6543210
  for
  urgent
  queries

Number of mappers selected: 2

----------------------------------------
             Mapping Phase
----------------------------------------
Splitting work among 2 mapper(s)...

Mapper 0 processing words:
  Sending word: "For" to mapper 0
  Sending word: "customer" to mapper 0
  Sending word: "support" to mapper 0
  Sending word: "please" to mapper 0
  Sending word: "call" to mapper 0
  Sending word: "123" to mapper 0
  Sending word: "4567890" to mapper 0
  Sending word: "or" to mapper 0
  Sending word: "8001234567" to mapper 0
  Sending word: "You" to mapper 0
```



```
Mapper 0 processing words:
  Sending word: "For" to mapper 0
  Sending word: "customer" to mapper 0
  Sending word: "support" to mapper 0
  Sending word: "please" to mapper 0
  Sending word: "call" to mapper 0
  Sending word: "123" to mapper 0
  Sending word: "4567890" to mapper 0
  Sending word: "or" to mapper 0
  Sending word: "8001234567" to mapper 0
  Sending word: "You" to mapper 0
  Sending word: "can" to mapper 0

Mapper 1 processing words:
  Sending word: "also" to mapper 1
  Sending word: "send" to mapper 1
  Sending word: "a" to mapper 1
  Sending word: "text" to mapper 1
  Sending word: "message" to mapper 1
  Sending word: "to" to mapper 1
  Sending word: "987" to mapper 1
  Sending word: "6543210" to mapper 1
  Sending word: "for" to mapper 1
  Sending word: "urgent" to mapper 1
  Sending word: "queries" to mapper 1

Waiting for mappers to finish...

Collecting results from Mapper 0...
  Received intermediate pair: ("For", 1)
  Received intermediate pair: ("customer", 1)
  Received intermediate pair: ("support", 1)
  Received intermediate pair: ("please", 1)
  Received intermediate pair: ("call", 1)
  Received intermediate pair: ("123", 1)
  Received intermediate pair: ("4567890", 1)
  Received intermediate pair: ("or", 1)
  Received intermediate pair: ("8001234567", 1)
  Received intermediate pair: ("You", 1)
  Received intermediate pair: ("can", 1)

Collecting results from Mapper 1...
  Received intermediate pair: ("also", 1)
  Received intermediate pair: ("send", 1)
  Received intermediate pair: ("a", 1)
  Received intermediate pair: ("text", 1)
  Received intermediate pair: ("message", 1)
  Received intermediate pair: ("to", 1)
  Received intermediate pair: ("987", 1)
  Received intermediate pair: ("6543210", 1)
  Received intermediate pair: ("for", 1)
  Received intermediate pair: ("urgent", 1)
  Received intermediate pair: ("queries", 1)

----------------------------------------
```



```
  ("987", 1)
  ("For", 1)
  ("You", 1)
  ("a", 1)
  ("also", 1)
  ("call", 1)
  ("can", 1)
  ("customer", 1)
  ("for", 1)
  ("message", 1)
  ("or", 1)
  ("please", 1)
  ("queries", 1)
  ("send", 1)
  ("support", 1)
  ("text", 1)
  ("to", 1)
  ("urgent", 1)

----------------------------------------
             Reducing Phase
----------------------------------------
Aggregating counts for each unique word...
  123 : 1
  4567890 : 1
  6543210 : 1
  8001234567 : 1
  987 : 1
  For : 1
  You : 1
  a : 1
  also : 1
  call : 1
  can : 1
  customer : 1
  for : 1
  message : 1
  or : 1
  please : 1
  queries : 1
  send : 1
  support : 1
  text : 1
  to : 1
  urgent : 1

----------------------------------------
           Final Word Counts
----------------------------------------
(Listed above are the aggregated results.)

Processing complete. All reducers have finished.
Thank you for using this MapReduce simulation!

aqsa@aqsa-ThinkPad-Yoga-260:~/Documents/5th sem/os/project$ S
```

Results and Observations:

- The program accurately counts words from the input text.
- Multiple mappers and reducers enhance performance, especially for large datasets.
- Dynamic mapper allocation ensures efficient resource utilization.

Design Advantages:

1. **Parallelism**:
   - Mappers and reducers work in parallel for faster processing.
2. **Scalability**:
   - The program dynamically adjusts the number of mappers based on input size.
3. **Thread-Safe Shared Data**:
   - Mutex ensures no race conditions during intermediate data insertion.
4. **Modularity**:
   - Clear separation of input handling, mapping, shuffling, and reducing phases.

Improvements:

1. **Dynamic Memory Management**:
   - Replace fixed-size arrays with dynamic data structures for larger datasets.
2. **Error Handling**:
   - Improve error messages and recovery mechanisms for pipe and thread failures.
3. **Output Formatting**:
   - Save results to a file or allow for customizable output formats.

Conclusion:

This MapReduce simulation effectively demonstrates the core principles of distributed data processing. By leveraging processes and threads, the program achieves parallelism and handles input dynamically, making it suitable for processing large datasets.