Name: Kushal Kishor Shankhapal

Group A, Assignment 3: Document Retrieval

Problem Statement:

To implement a program for Retrieval of documents using inverted files

document_retrieval.py

```
import re
from collections import defaultdict

def preprocess(text):
    # Simple tokenizer: lowercasing and split by non-alphanumeric chars
    return re.findall(r'\b\w+\b', text.lower())

def build_inverted_index(documents):
    # documents: dict {doc_id: text}
    inverted_index = defaultdict(lambda: defaultdict(list))
    # word -> {doc_id: [positions]}
    for doc_id, text in documents.items():
        words = preprocess(text)
        for pos, word in enumerate(words):
            inverted_index[word][doc_id].append(pos)
    return inverted_index

def search_single_word(inverted_index, word):
    word = word.lower()
    if word in inverted_index:
        return inverted_index[word]
    else:
        return {}

def search_phrase(inverted_index, phrase, documents):
    words = preprocess(phrase)
    if not words:
        return {}

    # Get occurrence dict for each word
    occurrence_dicts = [inverted_index.get(w, {}) for w in words]
    if any(not d for d in occurrence_dicts):
        # If any word not found, phrase not found
        return {}

    # Find docs containing all words
    common_docs = set(occurrence_dicts[0].keys())
    for d in occurrence_dicts[1:]:
        common_docs &= d.keys()
    if not common_docs:
        return {}

    result = {}
    # For each common doc, check if words appear in sequence
    for doc in common_docs:
        positions_lists = [occurrence_dicts[i][doc] for i in range(len(words))]
        # Check for sequential positions
        for start_pos in positions_lists[0]:
            if all((start_pos + i) in positions_lists[i] for i in range(len(words))):
                result.setdefault(doc, []).append(start_pos)
    return result

def print_results(results, documents):
    if not results:
        print("No matches found.")
```

```
        return
    for doc, positions in results.items():
        print(f"Document ID: {doc}")
        for pos in positions:
            snippet = documents[doc].split()[pos:pos+5]
            print("...", " ".join(snippet), "...")
        print()

# Example usage
documents = {
    1: "This is a text. A text has many words. Words are made from letters.",
    2: "Letters are important in words and text."
}

index = build_inverted_index(documents)

print("This is our document: \n")
for lines in documents:
      print("Line " + str(lines) +  ": " + documents[lines] + "\n")

query_word = input("Enter a single word to search: ")
print(f"Search results for single word '{query_word}':")
results = search_single_word(index, query_word)
print_results(results, documents)

query_phrase = input("Enter phrase to search: ")
print(f"Search results for phrase '{query_phrase}':")
results = search_phrase(index, query_phrase, documents)
print_results(results, documents)
```

Output:

```
$ python3 Document_Retrieval.py
This is our document:

Line 1: This is a text. A text has many words. Words are made from letters.

Line 2: Letters are important in words and text.

Enter a single word to search: text
Search results for single word 'text':
Document ID: 1
... text. A text has many ...
... text has many words. Words ...

Document ID: 2
... text. ...

Enter phrase to search: a text
Search results for phrase 'a text':
Document ID: 1
... a text. A text has ...
... A text has many words. ...
```