Nama : Raihan Rahmanda Junianto

NIM : 222112303

Kelas : 3SD2

Responsi Pertemuan 3 Praktikum Information Retrieval

---

1. **Menggunakan sekumpulan dokumen pada folder 'berita', setelah dilakukan preprocessing pada penugasan Modul 2, tambahkan kode untuk menghasilkan inverted index dengan output berupa term dan daftar lokasinya (posting lists).**

   Berdasarkan soal di atas, dibuatlah kode sebagai berikut.

```python
import os
from spacy.lang.id import Indonesian
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
from spacy.lang.id.stop_words import STOP_WORDS

nlp = Indonesian()

path = "D:/RAIHAN STIS/Perkuliahan/SEMESTER 5/Praktikum INFORMATION RETRIEVAL/Pertemuan (2)/berita"

def read_text_file(file_path):
    with open(file_path, 'r') as f:
        content = f.read()
    return content

def preprocess_text(text):
    stemmer = StemmerFactory().create_stemmer()
    stemmed_text = stemmer.stem(text)

    doc = nlp(stemmed_text)
    tokens = [token.text for token in doc if token.text.lower() not in STOP_WORDS]

    return tokens


inverted_index = {}

for file in os.listdir(path):
    if os.path.isfile(os.path.join(path, file)) and file.endswith(".txt"):
        file_path = os.path.join(path, file)

        text = read_text_file(file_path)

        cleaned_tokens = preprocess_text(text)

        # Update the inverted index
        for token in set(cleaned_tokens):  # Use set to avoid duplicate documents
            inverted_index.setdefault(token, []).append(file)

for term, documents in inverted_index.items():
    print(f"Term: {term}")
    print(f"Document yang mengandung term tersebut: {', '.join(documents)}")
    print("-" * 100)
```

Sama seperti sebelumnya, kode kali ini menggunakan library yang sama untuk melakukan preprocessing teks, yaitu Spacy dan Sastrawi. Pada program kali ini,

preprocessing teks dilakukan secara sekaligus pada satu kesatuan fungsi 'preprocess_text(text)', seperti yang terlihat di bawah ini.

```
def preprocess_text(text):
    stemmer = StemmerFactory().create_stemmer()
    stemmed_text = stemmer.stem(text)

    doc = nlp(stemmed_text)
    tokens = [token.text for token in doc if token.text.lower() not in STOP_WORDS]

    return tokens
```

Pada fungsi tersebut, stemming dilakukan terlebih dahulu dan di akhiri dengan eliminasi stopword serta tokenisasi. Oleh karena itu, output dari fungsi tersebut berupa list (token) dari teks yang sudah bersih.

Selanjutnya program berlanjut ke dalam pembuatan inverted index seperti yang terlihat pada kode program di bawah ini.

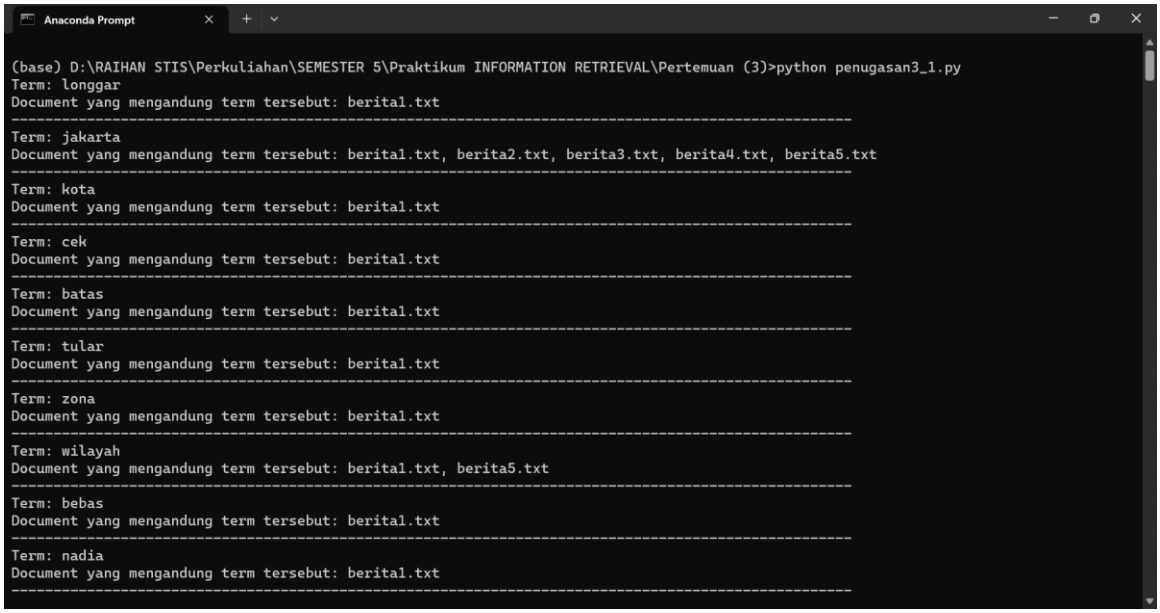
```
inverted_index = {}

for file in os.listdir(path):
    if os.path.isfile(os.path.join(path, file)) and file.endswith(".txt"):
        file_path = os.path.join(path, file)

        text = read_text_file(file_path)

        cleaned_tokens = preprocess_text(text)

        # Update the inverted index
        for token in set(cleaned_tokens):  # Use set to avoid duplicate documents
            inverted_index.setdefault(token, []).append(file)
```

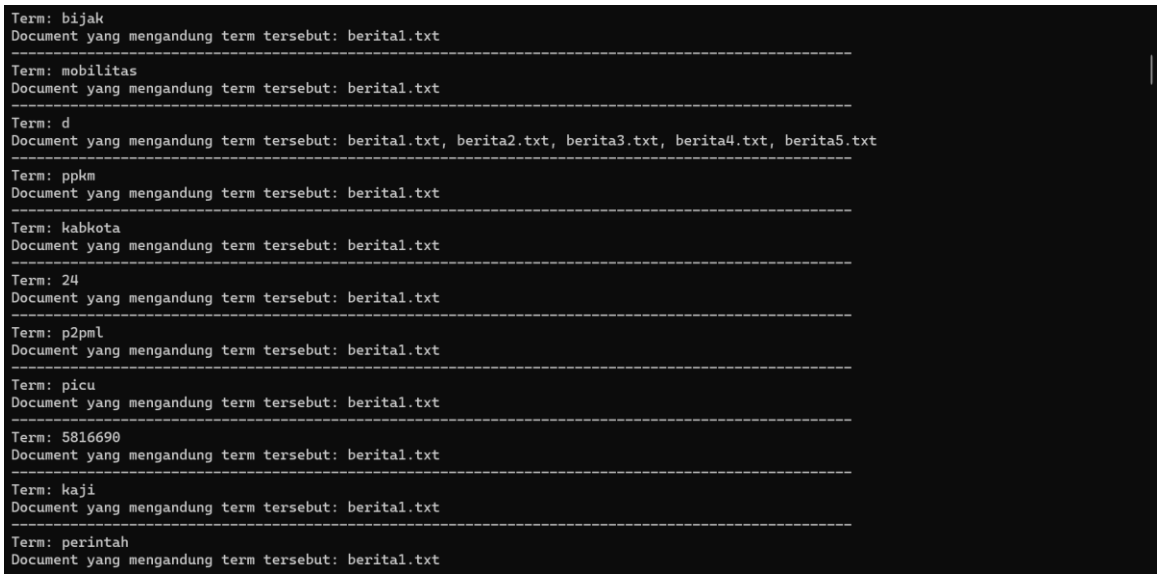
Pembuatan inverted index pada penggalan kode program di atas diawali dengan menginisiasi variabel inverted_index dengan struktur yang kosong. Selanjutnya, dilakukan perulangan (looping) semua file di dalam direktori yang dimaksud menggunakan sintaks 'for file in os.listdir(path)'. Kemudian, terdapat pengecekan kondisi untuk mengetahui bahwa file yang akan diproses hanya yang berekstensi .txt saja. Pada tahap selanjutnya, terdapat pemanggilan fungsi 'read_text_file' serta fungsi 'preprocess_text' dan memasukkannya ke dalam variabel 'cleaned_tokens' untuk menyimpan token-token hasil preprocessing.

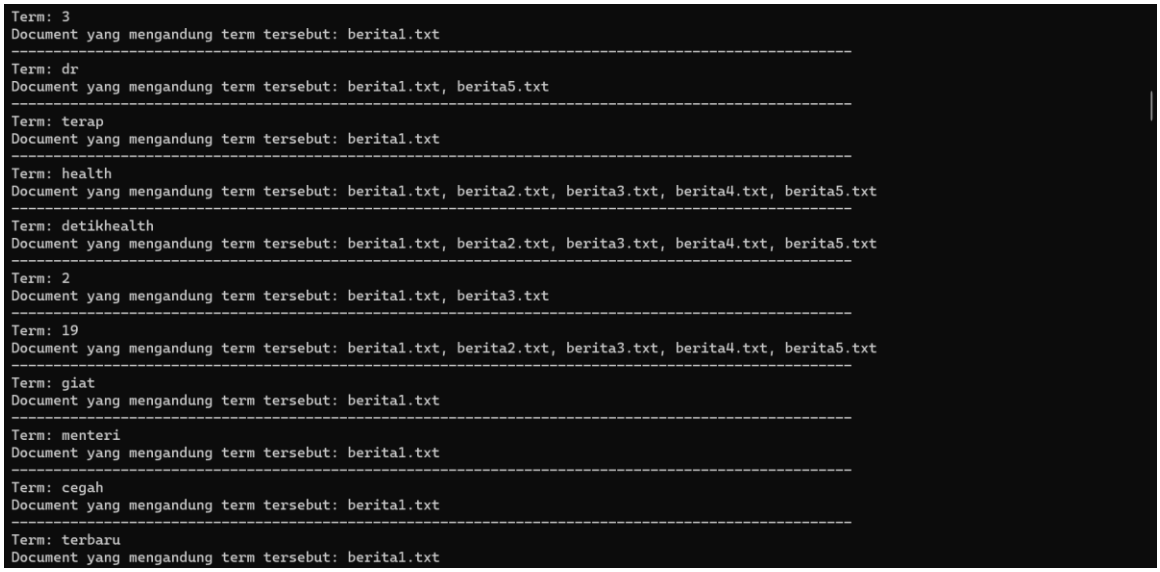
Pada tahap terakhir, terdapat proses untuk mengupdate nilai dari variabel 'inverted_index'. Pada iterasi 'for token in set(cleaned_tokens)', menggunakan fungsi 'set' untuk menghilangkan duplikasi dari token-token yang terdapat di suatu dokumen. Kemudian, inverted index diupdate dengan sintaks 'inverted_index.setdefault(token,

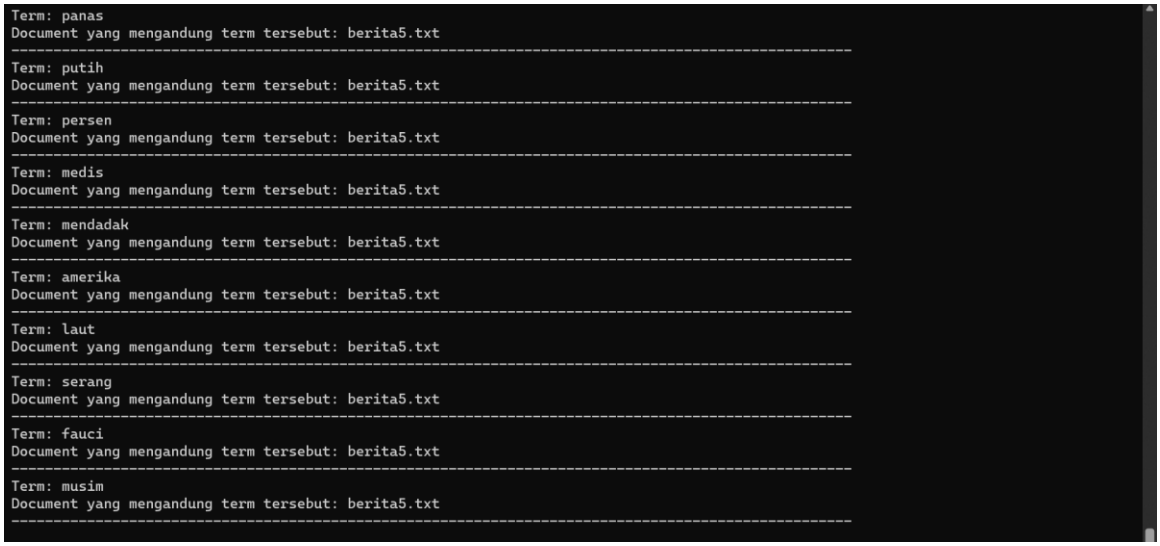
[]).append(file)’. Proses tersebut menggunakan fungsi ‘setdefault’ untuk menambahkan entri baru ke dalam kamus atau mengambil entri yang sudah ada jika token tersebut telah ada dalam kamus. Artinya, jika token sudah tersedia dalam variabel dict ‘inverted_index’, maka perintah tersebut akan mengambil daftar dokumen yang mengandung token tersebut, dan menambahkan file ke daftar tersebut. Namun, jika token belum tersedia dalam variabel dict ‘inverted_index’, maka perintah tersebut akan memebuat entri baru dengan token sebagai kunci dan menginisialisasi nilai untuk kunci tersebut sebagai sebuat daftar kosong ([]). Kemudian, perintah .append(file) akan menambahkan file ke daftar yang baru dibuat ini.

Dengan demikian, berikut output yang dihasilkan oleh kode program di atas.









2. Kemudian tambahkan kode untuk melakukan boolean retrieval dari inverted index pada Penugasan 1. Perhatikan daftar dokumen yang dikembalikan ketika menuliskan query berikut.

Berikut tambahan kode program untuk melakukan Boolean retrieval dari inverted index.

```python
def AND(posting1, posting2=None, posting3=None):
    p1, p2, p3 = 0, 0, 0
    result = []

    if posting2 is None and posting3 is None:
        while p1 < len(posting1):
            result.append(posting1[p1])
            p1 += 1

    elif posting3 is None:
        while p1 < len(posting1) and p2 < len(posting2):
            if posting1[p1] == posting2[p2]:
                result.append(posting1[p1])
                p1 += 1
                p2 += 1
            elif posting1[p1] > posting2[p2]:
                p2 += 1
            else:
                p1 += 1
    else:
        while p1 < len(posting1) and p2 < len(posting2) and p3 < len(posting3):
            if posting1[p1] == posting2[p2] == posting3[p3]:
                result.append(posting1[p1])
                p1 += 1
                p2 += 1
                p3 += 1
            else:
                min_val = min(posting1[p1], posting2[p2], posting3[p3])
                if posting1[p1] == min_val:
                    p1 += 1
                if posting2[p2] == min_val:
                    p2 += 1
                if posting3[p3] == min_val:
                    p3 += 1

    return result

def OR(posting1, posting2):
    p1 = 0
    p2 = 0
    result = list()
    while p1 < len(posting1) and p2 < len(posting2):
        if posting1[p1] == posting2[p2]:
            result.append(posting1[p1])
            p1 += 1
            p2 += 1
        elif posting1[p1] > posting2[p2]:
            result.append(posting2[p2])
            p2 += 1
        else:
            result.append(posting1[p1])
            p1 += 1
    while p1 < len(posting1):
        result.append(posting1[p1])
        p1 += 1
    while p2 < len(posting2):
        result.append(posting2[p2])
        p2 += 1
    return result

def NOT(term, inverted_index):
    matching_documents = set()

    if term in inverted_index:
        matching_documents.update(inverted_index[term])

    all_documents = set()
    for documents in inverted_index.values():
        all_documents.update(documents)

    not_matching_documents = all_documents - matching_documents

    result = sorted(not_matching_documents)

    return result

hasil1 = AND(inverted_index['corona'])
print("term 'corona' berada pada file:", hasil1)

hasil2 = AND(inverted_index['covid'])
print("term 'covid' berada pada file:", hasil2)

hasil3 = AND(inverted_index['vaksin'])
print("term 'vaksin' berada pada file:", hasil3)

hasil4 = OR(inverted_index['corona'],inverted_index['covid'])
print("term 'corona' OR 'covid' berada pada file:", hasil4)

hasil5 = AND(inverted_index['vaksin'],inverted_index['corona'])
print("term 'vaksin' AND 'corona' berada pada file", hasil5)

hasil6 = AND(inverted_index['vaksin'],inverted_index['corona'],inverted_index['pfizer'])
print("term 'vaksin' AND 'corona' AND 'pfizer' berada pada file", hasil6)

hasil7 = NOT('vaksin', inverted_index)
print(f"Term 'vaksin' tidak berada pada file: {hasil7}")
```

Pada kasus ini, boolean retrieval dilakukan menggunakan tiga fungsi yaitu AND, OR, dan NOT. Berikut penjelasan mengenai ketiga fungsi tersebut.

- Fungsi AND

  Fungsi kali ini telah sedikit dimodifikasi sehingga dapat menerima satu, dua, atau tiga argument. Variabel p1, p2, p3 diinisialisasi dengan nilai 0 dan variabel 'result' sebagai daftar kosong.

```
def AND(posting1, posting2=None, posting3=None):
    p1, p2, p3 = 0, 0, 0
    result = []
```

  a. Pada kasus satu argument, hasil operasi AND adalah posting list itu sendiri, dan akan menyalin semua elemen dari posting1 ke dalam variabel 'result'.

```
if posting2 is None and posting3 is None:
    while p1 < len(posting1):
        result.append(posting1[p1])
        p1 += 1
```

  b. Pada kasus dua argument, akan dilakukan suatu iterasi serta pengecekan beberapa kondisi. Pada saat posisi posting1 (yang ditunjukkan dengan indeks p1) sama dengan posisi posting2 (yang ditunjukkan dengan indeks p2), maka elemen tersebut akan ditambahkan ke dalam variabel 'result'. Namun, jika posisi saat ini dalam posting1 lebih besar dari posisi saat ini dalam posting2, p2 akan ditambahkan, menggerakkan pencarian ke dalam posting2. Sebaliknya, jika posisi saat ini dalam posting1 lebih kecil dari posisi saat ini dalam posting2, p1 akan ditambahkan, menggerakkan pencarian ke dalam posting1.

```
elif posting3 is None:
    while p1 < len(posting1) and p2 < len(posting2):
        if posting1[p1] == posting2[p2]:
            result.append(posting1[p1])
            p1 += 1
            p2 += 1
        elif posting1[p1] > posting2[p2]:
            p2 += 1
        else:
            p1 += 1
```

  c. Pada kasus tiga argument, akan dilakukan perbandingan posisi saat ini dalam semua posting list. Jika semua posisi saat ini sama, maka elemen ini adalah hasil dari operasi AND, dan akan ditambahkan ke result. Namun, jika tidak sama, maka akan dilkakukan penentuan nilai terkecil di antara ketiga posisi saat ini yang dimasukkan

ke dalam variabel ‘min_val’ dan akan menggerakkan posisi saat ini yang sesuai untuk mendekati nilai terkecil tersebut.

```
else:
    while p1 < len(posting1) and p2 < len(posting2) and p3 < len(posting3):
        if posting1[p1] == posting2[p2] == posting3[p3]:
            result.append(posting1[p1])
            p1 += 1
            p2 += 1
            p3 += 1
        else:
            min_val = min(posting1[p1], posting2[p2], posting3[p3])
            if posting1[p1] == min_val:
                p1 += 1
            if posting2[p2] == min_val:
                p2 += 1
            if posting3[p3] == min_val:
                p3 += 1
```

- Fungsi OR

  Pada fungsi ini terdapat tiga iterasi. Di dalam iterasi pertama ‘(while p1 < len(posting1) and p2 < len(posting2)), akan dilakukan pengecekan beberapa kondisi. Jika posisi saat ini dalam kedua posting list sama, elemen tersebut adalah hasil dari operasi OR, dan akan ditambahkan ke result. Kemudian, kedua indeks p1 dan p2 akan dilakukan proses increment. Namun, kika posisi saat ini dalam posting1 lebih besar dari posisi saat ini dalam posting2, maka elemen dari posting2 akan ditambahkan ke result, dan p2 akan dilakukan proses increment. Sama seperti sebelumnya, Jika posisi saat ini dalam posting1 lebih kecil dari posisi saat ini dalam posting2, maka elemen dari posting1 akan ditambahkan ke result, dan p1 akan dilakukan proses increment. Setelah kondisi pada iterasi pertama tidak tercukupi, maka elemen-elemen yang tersisi dari posting1 dan posting2 akan ditangi oleh dua iterasi lainnya dan hasilnya akan ditambahkan ke dalam variabel ‘result’.

- Fungsi NOT

  Fungsi ini akan mencari dokumen yang mengandung term tersebut dan menambahkannya ke dalam variabel ‘matching_document’. Selanjutnya, dengan iterasi, fungsi tersebut akan mengumpukan semua dokumen pada inverted index lalu menyimpannya ke variabel ‘all_documents’. Lalu, unutk mencari dokumen yang tidak mengandung term yang dimaksud dilakukan dengan cara mencari perbedaan antara ‘all_documents’ dan ‘matching_documents.’. Hasil dari proses tersebut di simpan ke dalam ‘not_matching_documents’ lalu dikembalikan ke dalam program melalui variabel result yang berisi variabel ‘‘not_matching_documents’ yang sudah terurut.

Berikut, output dari keseluruhan kode program di atas.

```
(base) D:\RAIHAN STIS\Perkuliahan\SEMESTER 5\Praktikum INFORMATION RETRIEVAL\Pertemuan (3)>python penugasan3_2.py
term 'corona' berada pada file: ['berita3.txt', 'berita5.txt']
term 'covid' berada pada file: ['berita1.txt', 'berita2.txt', 'berita3.txt', 'berita4.txt', 'berita5.txt']
term 'vaksin' berada pada file: ['berita2.txt', 'berita3.txt']
term 'corona' OR 'covid' berada pada file: ['berita1.txt', 'berita2.txt', 'berita3.txt', 'berita4.txt', 'berita5.txt']
term 'vaksin' AND 'corona' berada pada file ['berita3.txt']
term 'vaksin' AND 'corona' AND 'pfizer' berada pada file ['berita3.txt']
term 'vaksin' tidak berada pada file: ['berita1.txt', 'berita4.txt', 'berita5.txt']
```

3. **Modifikasi kode fungsi AND sehingga dapat melakukan optimasi query untuk list postings berikut:**

Berikut disajikan modifikasi dari fungsi AND.

```python
def AND_optimized(postings):
    if not postings:
        return []

    postings.sort(key=len)
    result = postings[0]

    for i in range(1, len(postings)):
        current_postings = postings[i]
        new_result = []

        ptr1, ptr2 = 0, 0

        while ptr1 < len(result) and ptr2 < len(current_postings):
            if result[ptr1] == current_postings[ptr2]:
                new_result.append(result[ptr1])
                ptr1 += 1
                ptr2 += 1
            elif result[ptr1] < current_postings[ptr2]:
                ptr1 += 1
            else:
                ptr2 += 1

        result = new_result
        if not result:
            break

    return result
```

Fungsi AND_optimized merupakan modifikasi dari fungsi AND sebelumnya sehingga dapat menerima beberapa posting sesuai keinginan pengguna. Pada awal fungsi, terdapat pengecekan terhadap isi dari postings. Selanjutnya, terdapat sintaks ‘postings.sort(key=len)’ untuk mengurutkan postings berdasarkan panjangnya. Hal tersebut bertujuan untuk mengurangi jumlah perbandingan yang perlu dilakukan. Pada bagian awal,

variabel ‘result’ diinisialisasi sebagai postings[0], yaitu posting yang memiliki length terpendek.

Selanjutnya, terdapat sintaks utama untuk melakukan operasi AND yang diawali dengan iterasi ‘for i in range(1, len(postings))’ sebanyak jumlah postings. Sama seperti sepelumnya, terdapat iterasi lainnya untuk membandingkan posisi dari masing-masing posting. Dalam loop ‘while ptr1 < len(result) and ptr2 < len(current_postings))’, program akan membandingkan posisi saat ini dalam ‘result’ dan ‘current_postings’. Jika posisi saat ini dalam keduanya sama, maka dokumen tersebut ada dalam kedua posting list, dan akan ditambahkan ke ‘new_result’. Selanjutnya, kedua indeks ptr1 dan ptr2 akan diincremenkan. Namun, jika posisi saat ini dalam result lebih kecil dari posisi saat ini dalam current_postings, ptr1 akan diincrementkan. Sebaliknya, Jika posisi saat ini dalam result lebih besar dari posisi saat ini dalam current_postings, maka ptr2 lah yang akan diincrementkan. Pada akhir iterasi while, variabel ‘new_result’ akan dimasukkan ke dalam ‘result’, lalu variabel ‘result’ tersebut akan dikembalikan ke main program.

Berikut tampilan keseluruhan kode program beserta outputnya.

```python
def AND_optimized(postings):
    if not postings:
        return []

    postings.sort(key=len)
    result = postings[0]

    for i in range(1, len(postings)):
        current_postings = postings[i]
        new_result = []

        ptr1, ptr2 = 0, 0

        while ptr1 < len(result) and ptr2 < len(current_postings):
            if result[ptr1] == current_postings[ptr2]:
                new_result.append(result[ptr1])
                ptr1 += 1
                ptr2 += 1
            elif result[ptr1] < current_postings[ptr2]:
                ptr1 += 1
            else:
                ptr2 += 1

        result = new_result
        if not result:
            break

    return result

search_terms = []
while True:
    try:
        num_terms = int(input("Masukkan jumlah term yang akan diinput: "))
        break
    except ValueError:
        print("Masukkan angka yang valid untuk jumlah term.")


for i in range(num_terms):
    search_term = input("Masukkan term pencarian ke-{0}: ".format(i + 1))
    search_terms.append(search_term.strip())

posting_lists = [inverted_index.get(term, []) for term in search_terms]

search_result = AND_optimized(posting_lists)

if search_result:
    print("Hasil pencarian untuk terms:", search_terms)
    for file in search_result:
        print("Term ditemukan pada file:", file)
else:
    print("Tidak ada hasil pencarian yang cocok dengan terms:", search_terms)
```

```
Anaconda Prompt

(base) D:\RAIHAN STIS\Perkuliahan\SEMESTER 5\Praktikum INFORMATION RETRIEVAL\Pertemuan (3)>python penugasan3_3.py
Masukkan jumlah term yang akan diinput: 3
Masukkan term pencarian ke-1: vaksin
Masukkan term pencarian ke-2: corona
Masukkan term pencarian ke-3: pfizer
Hasil pencarian untuk terms: ['vaksin', 'corona', 'pfizer']
Term ditemukan pada file: berita3.txt
```