LEMBAR KERJA PRAKTIKUM 5 STRUKTUR DATA

ITERATIF & REKURSIF DALAM STRUKTUR DATA

A. Tujuan Praktikum

Setelah mengikuti praktikum ini, mahasiswa mampu:

1. Mengimplementasikan algoritma secara iteratif dan rekursif.
2. Memahami peran **call stack** dalam rekursi.
3. Mengimplementasikan rekursi pada struktur data.

B. Implementasikan Program di bawah ini

```
#include <bits/stdc++.h>
using namespace std;
struct Node {
  int data;
  Node* next;
};

// Allocates a new node with given data
Node *newNode(int data)
{
  Node *new_node = new Node;
  new_node->data = data;
  new_node->next = NULL;
  return new_node;
}

// Function to insert a new node at the
// end of linked list using recursion.
Node* insertEnd(Node* head, int data)
{
  // If linked list is empty, create a
  // new node (Assuming newNode() allocates
  // a new node with given data)
  if (head == NULL)
     return newNode(data);

  // If we have not reached end, keep traversing
  // recursively.
  else
     head->next = insertEnd(head->next, data);
  return head;
```

```
}

void traverse(Node* head)
{
  if (head == NULL)
  return;

  // If head is not NULL, print current node
  // and recur for remaining list
  cout << head->data << " ";

  traverse(head->next);
}

// Driver code
int main()
{
  Node* head = NULL;
  head = insertEnd(head, 6);
  head = insertEnd(head, 8);
  head = insertEnd(head, 10);
  head = insertEnd(head, 12);
  head = insertEnd(head, 14);
  traverse(head);
}
```

C. Berdasarkan program di atas, lakukan proses tracing, dan watch variables untuk melihat alur kerja dari program tersebut. Setelah memahami alur program, diskusikan hal-hal berikut ini dengan asisten dan teman-teman Anda (hasil diskusi tidak perlu dituliskan).

1. Lakukan penelusuran terhadap eksekusi statement: head = insertEnd(head, 14);
   Diskusikan:
   - Urutan pemanggilan fungsi
   - Nilai parameter head pada setiap pemanggilan
   - Kapan node baru dibuat
   - Bagaimana proses *unwinding* terjadi hingga kembali ke main()
2. Berapa jumlah stack frame yang terbentuk saat menambahkan 14?
3. Apa kompleksitas waktu dan ruang dari insertEnd()? Jelaskan berdasarkan hasil tracing.
4. Lakukan penelusuran terhadap eksekusi statement: traverse(head);
   Diskusikan:
   - Urutan pemanggilan fungsi
   - Isi call stack
   - Urutan output yang dihasilkan

5. Mengapa urutan output sesuai dengan urutan node pada linked list?
6. Apakah traverse() juga memiliki kompleksitas ruang O(n)? Mengapa?
7. Mengapa linked list cocok diimplementasikan secara rekursif?
8. Jika jumlah node sangat besar (misalnya 100.000), apa risiko penggunaan rekursi?

D. Video dan Hasil Diskusi (dikumpulkan selambatnya satu minggu setelah praktikum ini)

Setelah Anda mempelajari program di atas, kerjakan tugas di bawah ini dengan kelompok Anda, lalu buat video diskusinya dan tuliskan hasil diskusinya (PDF), masukkan ke gdrive, lalu kumpulkan link gdrive pada form berikut: https://forms.gle/JCWa8XJy9n7Vnc7r7

1. Implementasi Fungsi Rekursif Tambahan:

(a) Insert Setelah Key

Implementasikan fungsi berikut secara rekursif:

```
void insertAfterKey(Node* p, int key, int data)
{
  ...
}
```

Ketentuan:

- Jika key ditemukan → tambahkan node setelahnya.
- Jika key tidak ditemukan → tambahkan node di akhir linked list.

Tuliskan hasil implementasi Anda.

(b) Hapus Node Berdasarkan Key

Implementasikan fungsi berikut secara rekursif:

```
void deleteKey(Node* prev, Node* p, int key)
{
  // prev menunjuk node sebelum node yang akan dihapus
  // p menunjuk node yang sedang diperiksa
  ...
}
```

Tuliskan hasil implementasi Anda.

2. Pengujian Program

Gunakan fungsi main() berikut:

```
int main()
{
        Node* head = NULL;
```

```
    head = insertEnd(head, 6);
    head = insertEnd(head, 8);
    head = insertEnd(head, 10);
    head = insertEnd(head, 12);
    head = insertEnd(head, 14);
    traverse(head);
    insertAfterKey(head, 12, 13);
    insertAfterKey(head, 8, 9);
    insertAfterKey(head, 15, 16);
    cout << "\n";
    traverse(head);
    cout << "\n";
    deleteKey(head, head, 12);
    traverse(head);
    return 0;
}
```

Tuliskan:

- Hasil output program
- Penjelasan perubahan struktur linked list pada setiap tahap

3. Tracing Lanjutan

Lakukan penelusuran detail terhadap:

1. insertAfterKey(head, 8, 9)
2. deleteKey(head, head, 12)

Tuliskan:

- Urutan pemanggilan fungsi
- Perubahan pointer prev dan p
- Proses penghapusan node
- Proses *unwinding* rekursi