E. LATIHAN

1. Buatlah program rekursif untuk menghitung segitiga Pascal !
   Syntax:

```
import 'dart:io';

void main() {
  int n = 5;
  for (int i = 0; i < n; i++) {
    for (int j = 0; j <= i; j++) {
      stdout.write('${pascal(i, j)} ');
    }
    print('');
  }
}
int pascal(int row, int col) {
  if (col == 0 || col == row) {
    return 1;
  }
  return pascal(row - 1, col - 1) + pascal(row - 1, col);
}
```

   Disisni kita per untuk menabah kan import ‘dart.io’ untuk menggunakan stdout.write, stdout.write(‘${pascal (I, j)} ‘);digunakan untuk mencetak nilai tanpa menambahkan newline setelahnya. Ini memungkinkan kita untuk mencetak semua nilai dalam satu baris dengan spasi sebagai pemisah.
   Output:

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1

Exited.
```

2. Buatlah program secara rekursif, masukkan jumlah N karakter dan cetak dalam semua kombinasi !
   Syntax:

```
void main() {
  int n = 3;
  List<String> characters = ['a', 'b', 'c'];
  List<String> result = [];
  generateCombinations(characters, '', n, result);
  print(result.join(' '));
```

```
}
void generateCombinations(List<String> characters, String current, int n,
List<String> result) {
  if (current.length == n) {
    result.add(current);
    return;
  }

  for (String char in characters) {
    generateCombinations(characters, current + char, n, result);
  }
}
```

'Main()' untuk menjumlahnkan karakter yang di jumlah kan, dan membuat list 'result' untuk menyimoan semua kombinasi yang dihasilkan,

Fungsi 'generatecombination' untuk memulai proses rekursif,

Output:

```
aaa aab aac aba abb abc aca acb acc baa bab bac bba bbb bbc bca bcb bcc caa cab cac cba cbb cbc
cca ccb ccc

Exited.
```

3. Buat program BinarySearch dengan Rekursif ! (data tentukan sendiri):
   Syntax:

```
void main() {
  List<int> data = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19];
  int target = 11;
  int result = binarySearch(data, target, 0, data.length - 1);
  if (result != -1) {
    print('Elemen $target ditemukan pada indeks: $result');
  } else {
    print('Elemen $target tidak ditemukan dalam array.');
  }
}
int binarySearch(List<int> data, int target, int left, int right) {

  if (left > right) {
    return -1;
  }
  int mid = left + (right - left) ~/ 2;
  if (data[mid] == target) {
```

```
    return mid;
  }
  if (target < data[mid]) {
    return binarySearch(data, target, left, mid - 1);
  }
  return binarySearch(data, target, mid + 1, right);
}
```

Disini mendefiniskan ‘targerr, yaitu elemen yang ingin kita cari dalam list,
Dan fungsi ‘ binarysearch’dipanggil dengan parameter list ‘data’ ‘targer’,indexs kiri (‘left’),
Dan indeks kanan (‘right’).
Output:

```
PS C:\struktur data dart> dart run L3.dart
Elemen 11 ditemukan pada indeks: 5
PS C:\struktur data dart> dart run L3.dart
Elemen 50 tidak ditemukan dalam array.
PS C:\struktur data dart>
```

4. Buatlah program rekursif untuk memecahkan permasalahan Menara Hanoi !
   Syntax:

```
void menaraHanoi(int n, String from, String to, String temp) {
  if (n == 1) {
    print("pindahkan disc 1 dari pasak $from ke pasak $to");
    return;
  }
//memindahkan submenara dengan (n-1) dari pasak "form" ke "temp"
// dengan bantuan pasak "to"
  menaraHanoi(n - 1, from, to, temp);

  print("pindahkan dics $n dari pasak $from ke pasak $to");
//memindahkan pasak dengan (n-1) dari pasak "from" ke "to" dengan
//bantuan pasak "temp"
  menaraHanoi(n - 1, from, temp, to);
}

void main() {
  int jumlahDisk = 3 ; //jumlah disk sesuai keinginan
  print("langkah-langkah memindahkan $jumlahDisk disk pada menara Hanoi:");
  menaraHanoi(jumlahDisk, "A", "B", "C");
}
```

Program ini merupakan program untuk menampilkan pergerakkan menara hanoi, yang merujuk pada class menaraHanoi. Secara umum algoritma menara hanoi, adalah memindahkan sub menara hanoi dengan n – 1 pin dari n pin ke tiang perantara. Lalu memindahkan pin ke n ke

tiang tujuan, lalu memindahkan sub menara hanoi dengan n – 1 pin yang ada di tiang perantara, ke tiang tujuan. StopCase nya jika n == 1. Jumlah disk : 3 Langkah-langkah nya adalah dengan :

a. Pindahkan disc 1 dari pasak A ke pasak C
b. Pindahkan disc 2 dari pasak A ke pasak B
c. Pindahkan disc 1 dari pasak C ke pasak B
d. Pindahkan disc 3 dari pasak A ke pasak C
e. Pindahkan disc 1 dari pasak B ke pasak A
f. Pindahkan disc 2 dari pasak B ke pasak C
g. Pindahkan disc 1 dari pasak A ke pasak C

5. Jelaskan proses rekursif untuk program dibawah ini !

```
void decToBin(int num) {
   if (num > 0) {
      decToBin(num ~/ 2);
      stdout.write('${num % 2}'); } }
```

penjelasan :
fungsi menerima parameter ‘num’ yang merupakan angka decimal yang dikonversi ke biner.
‘if (num > 0) ‘ disini fungsi akan melakukan konversi jika ‘num lebih besar dari 0, jika ‘num’ adlah 0 atau negative maka akan kosong
Nanti nya ‘decToBin(num ~/ 2);’ akan mamanggil recursif, dan dengan memnggil ‘decToBin(num ~/ 2);’ fungsi akan terus membagi ‘num’ dengan 2 hingga num menjadi 0.
Lalu ‘stdout.write('${num % 2}'); } }’ akan mencetak digit binner.

6. Jelaskan proses rekursif untuk program dibawah ini dengan memanggil test(“01101”, 4)!

```
int test(String s, int last) { if (last < 0) { return 0; } if (s[last] == "0") { return 2 * test(s, last - 1); }
return 1 + 2 * test(s, last - 1); }
```

penjelasan:
‘s’ String binner yang akan di proses.
‘last’ indek terakhir yang akandi periksa ‘s’
‘{ if (last < 0)’ jika ‘las’ kurang dari 0 maka fungsi aka mengembalikan
‘} if (s[last] == "0")’ jika indeks ‘last’ yaitu “0”, maka fungsi akan mengembalikannya 1 + 2 * test(s, last - 1); }

7. Jelaskan proses rekursif untuk program dibawah ini ! 1 2 3 4 5 6 7 8 9 10

```
bool search(List x, int size, int n) { if (size > 0) { if (x[size - 1] == n) { return true; } else { return search(x, size - 1, n); } }
return false; }
```

Penjelasan: ‘bool search(List x, int size, int n)’ disini list yang berisii elemen integer akan mencari ukuran dari ‘x’ yang mana jumlah nya akan diperiksa
‘{ if (x[size - 1] == n) { return true; }’ fungsi akan melanjutkan pencarian jika ‘size’ adalah 0, dan fungsi akan kembali 0,
‘else { return search(x, size - 1, n); } } return false; }’ jika fungsi tidak sama maka akan memnaggila dirinya sendiri dengan ‘search(List x, int size, int n)’ yang mana fungsi akan memeriksa elemen seblum list. Jika size ‘0’ maka fungsi mengembalikan ‘false’
Panggilan Pertama: search([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 10, 7)
size adalah 10, yang valid.

Memeriksa x[9] (yaitu 10) dengan n (yaitu 7). Tidak sama
Memanggil search([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 9, 7).

8. Jelaskan proses rekursif untuk program dibawah ini !

```
bool binarySearch(List<int> x, int start, int end, int n) {
  if (end < start) return false;
  int mid = (start + end) ~/ 2;

  if (x[mid] == n) {
    return true;
  } else {
    if (x[mid] < n) {
      return binarySearch(x, mid + 1, end, n);
    } else {
      return binarySearch(x, start, mid - 1, n);
    }
  }
}
```

Penjelasan :

Cek kondisi berhenti: Jika end < start, kembalikan false (tidak ditemukan).
Tentukan nilai tengah: mid = (start + end) ~/ 2.
Bandingkan dengan nilai yang dicari (n):

Jika x[mid] == n → Kembalikan true (ditemukan). Jika x[mid] < n → Cari di kanan (mid + 1 end). Jika x[mid] > n → Cari di kiri (start hingga mid - 1).

9. Jelaskan proses rekursif untuk program dibawah ini !

```
int f(int n) {
  if (n == 0) return 0;
  if (n == 1) return 1;
  if (n == 2) return 1;
  return 2 * f(n - 2) + f(n - 3);
}
```

Jika n == 0 → hasil 0

Jika n == 1 atau n == 2 → hasil 1

Jika n >= 3 → hasil dihitung dengan rumus:

2 * f(n - 2) + f(n - 3);

10. Jelaskan proses rekursif untuk program dibawah ini dengan memanggil square(5), cube(5), cube(123)?

Fungsi square(n) menghitung kuadrat (n^2) menggunakan rekursi.

Fungsi cube untuk menghitung kubus (n^3) menggunakan rekursi.

11. Ubahlah proses rekursif non tail menjadi tail rekursif untuk program dibawah ini !

```dart
import 'dart:io';

void decToBin(int num, String binStr) {
  if (num == 0) {
    stdout.write(binStr);
    return;
  }
  decToBin(num ~/ 2, '${num % 2}$binStr');
}

void main() {
  decToBin(10, '');
  stdout.writeln();
}
```

Disini perlu menambahkan para meter binStr untuk menyimpan binner secara bertahap

Dan menghindari operasi setelah rekursif, dan dimulai dengan string kosong (‘’) sebagai

Awalnya. Output:

```
PS C:\struktur data dart> dart run L11.dart
1010
PS C:\struktur data dart>
```

12. Ubahlah proses rekursif non tail menjadi tail rekursif untuk program dibawah ini !

Hasilnya:

```dart
bool gcdlike(int p, int q, bool result) {
  if (q == 0) {
    return (p == 1) && result;
  }
  return gcdlike(q, p % q, result);
}

void main() {
  print(gcdlike(35, 18, true));
  print(gcdlike(12, 18, true));
}
```

Penambahan parameter result digunakan untuk membawa hasil rekursif

Output:

```
PS C:\struktur data dart> dart run L12.dart
true
false
```

13. Ubahlah proses rekursif non tail menjadi tail rekursif untuk program dibawah ini !

Hasilnya:
```dart
int fTail(int n, int a, int b, int c) {
  if (n == 0) return a;
  if (n == 1) return b;
  if (n == 2) return c;
  return fTail(n - 1, b, c, 2 * b + a);
}

int f(int n) {
  return fTail(n, 0, 1, 1);
}

void main() {
  print(f(5));
  print(f(6));
  print(f(7));
}
```

Kita menghitung f(n-1) menggnukan hasil sebelumnya tanpa perlu menghitung ulang

14. Ubahlah proses rekursif non tail menjadi tail rekursif untuk program dibawah ini dengan memanggil square(5), cube(5), cube(123)?

Hasilnya:

```dart
int squareTail(int n, int acc) {
  if (n == 0) return acc;
  return squareTail(n - 1, acc + 2 * n - 1);
}

int square(int n) {
  return squareTail(n, 0);
}
```

```
void main() {
  print(square(5));
}
```

1. squareTail(n, acc): n: Angka yang ingin dihitung kuadratnya. acc: Akumulator untuk menyimpan hasil sementara.
2. Basis Rekursi: Jika n == 0, kembalikan nilai acc.
3. Rekursi: Kurangi n dan tambahkan hasil (2 * n - 1) ke akumulator.