คำถามตอบ

CH1

ข้อ 1: อธิบายว่าการแก้ปัญหาด้วยคอมพิวเตอร์ต่างจากการแก้ปัญหาในชีวิตประจำวันอย่างไร

คำตอบ: การแก้ปัญหาด้วยคอมพิวเตอร์มักจะต้องมีการระบุปัญหาอย่างชัดเจนและใช้ขั้นตอนวิธีที่เป็นระบบเพื่อหาคำตอบที่ถูกต้อง ในขณะที่การแก้ปัญหาในชีวิตประจำวันอาจรวมถึงการพึ่งพาประสบการณ์และการตัดสินใจอย่างเร็ว.

ข้อ 2: จงอธิบายความแตกต่างระหว่าง Heuristic และ Algorithmic solutions

คำตอบ: Heuristic solutions ใช้ประสบการณ์และการทดลองเพื่อหาคำตอบ ซึ่งอาจไม่ได้เป็นวิธีที่สมบูรณ์แบบแต่เพียงพอสำหรับการใช้งาน ในขณะที่ Algorithmic solutions ใช้ขั้นตอนวิธีที่ชัดเจนและเป็นระบบเพื่อหาคำตอบที่ถูกต้องและเชื่อถือได้.

ข้อ 3: อธิบายปัญหาที่เกิดขึ้นเมื่อโปรแกรมเครื่องคิดเลขไม่สามารถประมวลผลการหารด้วยศูนย์ได้

คำตอบ: ปัญหานี้เกิดขึ้นเนื่องจากการหารด้วยศูนย์ไม่มีคำตอบในทางคณิตศาสตร์ ทำให้โปรแกรมเครื่องคิดเลขต้องมีการจัดการข้อผิดพลาดนี้โดยการแจ้งเตือนหรือหยุดการประมวลผลเพื่อป้องกันความผิดพลาดและความเสียหายที่อาจตามมา.

ข้อ 4: อธิบายกระบวนการทั้งหกขั้นตอนในการแก้ปัญหาทางคอมพิวเตอร์ตามที่ได้กล่าวมาในเอกสาร

คำตอบ:

1. ระบุปัญหา
2. ทำความเข้าใจปัญหา
3. ระบุทางเลือกในการแก้ปัญหา
4. เลือกวิธีการที่ดีที่สุด
5. จัดทำรายการขั้นตอน
6. ประเมินผลของการแก้ปัญหา

ข้อ 5: จงอธิบายความยากลำบากที่อาจเกิดขึ้นในการแก้ปัญหาทางคอมพิวเตอร์

คำตอบ: ความยากลำบากในการแก้ปัญหาทางคอมพิวเตอร์อาจรวมถึงข้อจำกัดในการเข้าถึงข้อมูล, ความซับซ้อนของปัญหาที่สูง, การขาดทรัพยากร, และการไม่เข้าใจปัญหาอย่างถูกต้องซึ่งอาจนำไปสู่การเลือกวิธีการแก้ปัญหาที่ไม่เหมาะสมหรือไม่มีประสิทธิภาพ.

CH2

ข้อ 1: อธิบายว่าโครงสร้างการตัดสินใจ (Decision Structure) ในการเขียนโปรแกรมทำงานอย่างไร?

คำตอบ: โครงสร้างการตัดสินใจใช้ในการเลือกเส้นทางการทำงานของโปรแกรมตามเงื่อนไขที่กำหนด โดยมีสามแบบหลัก: if, if-else, และ if-else if-else. โครงสร้างนี้จะประเมินเงื่อนไข และเลือกทำงานตามเงื่อนไขที่เป็นจริง ช่วยให้โปรแกรมสามารถตอบสนองต่อสถานการณ์ต่างๆ ได้อย่างมีประสิทธิภาพ.

ข้อ 2: จงยกตัวอย่างโครงสร้างลูป (Loop Structure) และอธิบายประโยชน์ของมัน

คำตอบ: โครงสร้างลูปช่วยให้สามารถทำซ้ำกลุ่มคำสั่งบางอย่างได้หลายครั้ง ตัวอย่างเช่น โครงสร้าง for ใช้สำหรับทำซ้ำตามจำนวนครั้งที่ระบุ โครงสร้างลูปมีประโยชน์ในการทำงานกับข้อมูลจำนวนมาก หรือการประมวลผลที่ต้องทำซ้ำเดียวกันหลายครั้ง เช่น การคำนวณค่าในอาร์เรย์.

ข้อ 3: อธิบายการใช้งานของ Recursive Functions และข้อดีของการใช้งาน

คำตอบ: Recursive function คือฟังก์ชันที่เรียกใช้ตัวเองภายในการทำงานของมันเอง เหมาะสำหรับปัญหาที่สามารถแบ่งย่อยออกเป็นปัญหาย่อยๆ ที่คล้ายกัน เช่น การคำนวณ factorial ข้อดีของการใช้ Recursive คือ โค้ดที่เรียบง่ายและสามารถเข้าใจได้ง่าย เมื่อเทียบกับโค้ดที่ไม่ใช้การเรียกฟังก์ชันซ้ำ.

ข้อ 4: สร้าง IPO Chart สำหรับโปรแกรมคำนวณพื้นที่สี่เหลี่ยม โดยมีการรับค่าความยาวและความกว้างจากผู้ใช้

คำตอบ:

- **Input**: รับค่าความยาว (L) และความกว้าง (W)
- **Process**: คำนวณพื้นที่โดยใช้สูตร Area = L * W
- **Output**: แสดงผลพื้นที่ของสี่เหลี่ยม

ข้อ 5: อธิบายและแสดงตัวอย่างการใช้ Structure Chart ในการออกแบบโปรแกรม

คำตอบ: Structure Chart แสดงโครงสร้างโมดูลหรือฟังก์ชันของโปรแกรมและความสัมพันธ์ระหว่างพวกมัน ตัวอย่างเช่น โปรแกรมเครื่องคิดเลขอาจมีโมดูลสำหรับการรับข้อมูล, โมดูลคำนวณ, และโมดูลแสดงผล โดยแต่ละโมดูลทำงานร่วมกันเพื่อให้ได้ผลลัพธ์ที่ต้องการ ช่วยให้การออกแบบและการบำรุงรักษาโปรแกรมเป็นไปอย่างมีระเบียบและง่ายต่อการจัดการ.

CH3

ข้อ 1: อธิบายว่าอัลกอริทึมคืออะไร?

คำตอบ: อัลกอริทึมคือชุดของกฎหรือขั้นตอนที่กำหนดไว้เพื่อแก้ไขปัญหาคำนวณ ซึ่งอาจรวมถึงการเรียงลำดับ การค้นหา หรือการทำซ้ำของข้อมูล เพื่อให้ได้ผลลัพธ์ที่ต้องการจากข้อมูลนำเข้า.

ข้อ 2: อะไรคือ Big O Notation และมันใช้เพื่ออะไร?

คำตอบ: Big O Notation คือวิธีการที่ใช้เพื่ออธิบายความซับซ้อนของเวลาที่อัลกอริทึมต้องใช้ในการแก้ไขปัญหาขึ้นอยู่กับขนาดของข้อมูลนำเข้า. มันช่วยให้เราเข้าใจว่าอัลกอริทึมจะประมวลผลช้าหรือเร็วขึ้นเมื่อขนาดข้อมูลเพิ่มขึ้น.

ข้อ 3: ประเมินประสิทธิภาพของอัลกอริทึมที่ใช้เวลา $O(n^2)$ และ $O(\log n)$ เมื่อประมวลผลข้อมูลขนาดใหญ่

คำตอบ: อัลกอริทึมที่มีความซับซ้อน $O(n^2)$ จะใช้เวลามากขึ้นอย่างรวดเร็วเมื่อขนาดข้อมูลเพิ่มขึ้น เนื่องจากเวลาการทำงานเพิ่มขึ้นเป็นเท่าตัวของขนาดข้อมูล. ในทางตรงกันข้าม, อัลกอริทึมที่มีความซับซ้อน $O(\log n)$ จะมีการเพิ่มเวลาการทำงานที่น้อยมาก เมื่อขนาดข้อมูลเพิ่มขึ้น, เหมาะสำหรับข้อมูลขนาดใหญ่.

ข้อ 4: เขียนโปรแกรมฟังก์ชันที่คำนวณผลรวมของเลขคู่และเลขคี่จาก 1 ถึง N และให้แสดง Big O ของฟังก์ชันนี้

โค้ดตัวอย่าง:

```python
def sum_even_odd(N):
    sum_even = sum_odd = 0
    for i in range(1, N+1):
        if i % 2 == 0:
            sum_even += i
        else:
            sum_odd += i
    return sum_even, sum_odd

# Big O Notation: O(n)
```

ข้อ 5: อธิบายถึง trade-offs ระหว่างความซับซ้อนของเวลาและความจำในการเลือกใช้อัลกอริทึมต่างๆ

คำตอบ: การเลือกอัลกอริทึมสำหรับปัญหาใดๆ มักจะมี trade-offs ระหว่างความซับซ้อนของเวลาและความจำ. บางอัลกอริทึมอาจทำงานเร็วแต่ใช้ความจำเยอะ (เช่น แคชข้อมูล), ในขณะที่อัลกอริทึมอื่นอาจใช้เวลานานขึ้นแต่ใช้ความจำน้อยลง. การเลือกอัลกอริทึมที่เหมาะสมขึ้นอยู่กับข้อจำกัดและความต้องการเฉพาะของแต่ละปัญหา.

CH4

ข้อ 1: อธิบายความแตกต่างระหว่าง linked list และ array

คำตอบ: Linked list เป็นโครงสร้างข้อมูลที่ประกอบด้วย node ที่เชื่อมโยงกัน แต่ละ node มีสองส่วนคือ data และ pointer ที่ชี้ไปยัง node ถัดไป ข้อดีคือการเพิ่มหรือลบข้อมูลใดๆ สามารถทำได้ง่ายโดยไม่ต้องเปลี่ยนย้ายข้อมูลอื่นๆ ในโครงสร้าง แต่ข้อเสียคือการเข้าถึงข้อมูลต้องเริ่มจาก node แรกทุกครั้งทำให้ช้ากว่า array ที่สามารถเข้าถึงข้อมูลได้ทันทีด้วย index เนื่องจากข้อมูลถูกเก็บอย่างต่อเนื่องในหน่วยความจำ.

ข้อ 2: จงอธิบายประโยชน์ของการใช้ stack ในการเขียนโปรแกรม

คำตอบ: Stack เป็นโครงสร้างข้อมูลที่ใช้หลักการ Last In First Out (LIFO) ที่ประโยชน์สำคัญคือการจัดการกับการเรียกใช้ฟังก์ชันโดยจัดเก็บที่อยู่ของฟังก์ชันและตัวแปรท้องถิ่น ช่วยในการควบคุมการทำงานของโปรแกรมเมื่อมีการเรียกใช้ฟังก์ชันหลายชั้น และยังใช้ในการพัฒนาฟังก์ชันเช่น undo/redo ในแอปพลิเคชัน.

ข้อ 3: ให้คำจำกัดความของ Queue และอธิบายวิธีการทำงาน

คำตอบ: Queue เป็นโครงสร้างข้อมูลที่ทำงานตามหลักการ First In First Out (FIFO) ใช้ในสถานการณ์ที่ต้องการการจัดการข้อมูลหรือร้องขอในลำดับที่มาถึง มักใช้ในการจัดสรรทรัพยากรและจัดการกระบวนการในระบบปฏิบัติการ ช่วยให้สามารถจัดการกับข้อมูลหรือร้องขอที่ต้องได้รับการประมวลผลตามลำดับที่มาถึง.

ข้อ 4: อธิบายการใช้งานและประโยชน์ของ Circular Queue

คำตอบ: Circular Queue เป็นรูปแบบพิเศษของ queue ที่ตำแหน่งสุดท้ายของ queue จะเชื่อมต่อกับตำแหน่งแรก สร้างวงกลม มีประโยชน์ในการใช้ทรัพยากรหน่วยความจำอย่างมีประสิทธิภาพเพราะมันช่วยให้สามารถใช้ทรัพยากรที่มีอยู่ได้โดยไม่สูญเปล่า เหมาะสำหรับการใช้งานที่มีการหมุนเวียนข้อมูลหรืองานที่ต้องการการประมวลผลแบบวนซ้ำ.

ข้อ 5: อธิบายแนวคิดของ Priority Queue และการใช้งานในโปรแกรม

คำตอบ: Priority Queue เป็นรูปแบบของ queue ที่แต่ละองค์ประกอบมีค่าความสำคัญหรือลำดับความสำคัญ การลบข้อมูลจาก Priority Queue จะลบข้อมูลที่มีความสำคัญหรือลำดับสูงสุดออกก่อน มีประโยชน์ในการจัดการงานหรือข้อมูลที่มีความสำคัญต่างกัน เช่น การจัดสรร CPU ในระบบปฏิบัติการ หรือการจัดการร้องขอในระบบแบบเรียลไทม์.