Emp(eid: integer, ename: varchar, sal: integer, age: integer, did: integer)
Dept(did: integer, budget: integer, floor: integer, mgr eid: integer)

Salaries range from \$10,000 to \$100,000, ages vary from 20 to 80, each department has about five employees on average, there are 10 floors, and budgets vary from \$10,000 to \$1 million. You can assume uniform distributions of values.
For each of the following queries, which of the listed index choices would you choose to speed up the query?
Query: Print ename, age, and sal for all employees.

a. Clustered hash index on <ename, age, sal> fields of Emp.
b. Unclustered hash index on <ename, age, sal> fields of Emp.
c. Clustered B+ tree index on <ename, age, sal> fields of Emp.
d. Unclustered hash index on <eid, did> fields of Emp.
e. No index.

1. **(e) No index**. Since we want to print data of **all** employees.

Emp(eid: integer, ename: varchar, sal: integer, age: integer, did: integer)
Dept(did: integer, budget: integer, floor: integer, mgr eid: integer)

Salaries range from \$10,000 to \$100,000, ages vary from 20 to 80, each department has about five employees on average, there are 10 floors, and budgets vary from \$10,000 to \$1 million. You can assume uniform distributions of values.
For each of the following queries, which of the listed index choices would you choose to speed up the query?

Query: Find the dids of departments that are on the 10th floor and have a budget of less than \$15,000.

a. Clustered hash index on the floor field of Dept.
b. Unclustered hash index on the floor field of Dept.
c. Clustered B+ tree index on <floor, budget> fields of Dept.
d. Clustered B+ tree index on the budget field of Dept.
e. No index.

2. **(c) Clustered B+ tree index on <floor, budget> field of Dept**. I choose Clustered B+ tree because the result is range result and we want 10th floor less then $15,000 budget, then index on <floor, budget> will sort the data for us to easily query.

From the SQL query below, which index structure will not be beneficial to the query execution?

```
Select * From Apply, College
Where Apply.cName = College.cName
And Apply.major = 'cs' and College.enrollment <5000
```

a. Tree-based index on Apply.cName
b. Hash-baed index on Apply.major
c. Hash-based index on College.enrollment
d. Hash-based index on College.cName
e. More than one choices are not beneficial

3. **(c) hash-based index on College.enrollment** because hash will store data unordered. So it would be a good candidate only if we have equality selection.

สำหรับฐานข้อมูลที่ใช้ Unclustered hash index

- Leaf page แต่ละ page จะมีจำนวน Data entry เท่าใด
- ถ้าต้องการ List data record ออกมาทั้งหมดโดยใช้ Hash index ต้องใช้เวลาทั้งหมดเท่าใด

[กำหนดให้ occupancy rate = 67% (ไม่มี overflow chains), Data entry size = 20% ของ data records, B = จำนวน Data page เมื่อมี record แพ็คอยู่เต็มโดยไม่มี slot ว่าง, R = จำนวน record ในแต่ละ page, D = เวลาเฉลี่ยในการอ่าน/เขียนแต่ละ disk page]

4. 1. Since Data entry size = 20% of data records. 1 Page can contain $\frac{R}{0.20}$ data entries. Moreover, occupancy rate = 67% means 1 page contains only 67%. So, each leaf page contains $(\frac{67}{100})\frac{R}{0.20} = \frac{67}{20}R$ data entries.

   2. $BRD + \frac{BRD}{\frac{67}{20}R} = BRD + \frac{20}{67}BD$ where BRD is time for fetch data record and $\frac{20}{67}BD$ is time for accessing data entries

ในกรณีที่ใช้ Clustered files index (B+ tree: Alternative (1)) กับฐานข้อมูลที่มีการเพิ่ม record ใหม่อยู่ตลอด จึงกำหนดให้ occupancy rate เป็น 25% ให้หาค่า cost ในการหา leaf page ที่เหมาะสมในการ Insert ข้อมูลใหม่ในแต่ละครั้ง (Average case) [กำหนดให้ F = Fanout ของ B+ tree, B = จำนวน Data page เมื่อมี record แพ็คอยู่เต็มโดยไม่มี slot ว่าง, R = จำนวน record ในแต่ละ page, D = เวลาเฉลี่ยในการอ่าน/เขียนแต่ละ disk page]

5. Since occupancy rate = 25%, then number of data pages is $\frac{B}{0.25} = 4B$

   To find leaf page, the cost is $D \cdot height = D\log_F(4B)$