**UNIVERSITY OF COLOMBO, SRI LANKA**

**UNIVERSITY OF COLOMBO SCHOOL OF COMPUTING**

**DEGREE OF BACHELOR OF INFORMATION TECHNOLOGY (EXTERNAL)**

***Academic Year 2019 – 3rd Year Examination – Semester 6***

# *IT6405: Database Systems II*

**09th November, 2019**
***(TWO HOURS)***

**To be completed by the candidate**

BIT Examination Index No: .....

**Important Instructions:**

- The duration of the paper is **2 (two) hours**.
- The medium of instruction and questions is English.
- This paper has **4 questions** and **18 pages**.
- **Answer all questions.** All questions **carry** equal marks.
- **Write your answers** in English using the space provided **in this question paper**.
- Do not tear off any part of this answer book.
- Under no circumstances may this book, used or unused, be removed from the Examination Hall by a candidate.
- Note that questions appear on both sides of the paper.
  If a page is not printed, please inform the supervisor immediately.
- Calculators are **not** allowed.

**Questions Answered**

Indicate by a cross (×), (e.g. [1] ) the numbers of the questions answered.

| | Question numbers | | | | |
|---|---|---|---|---|---|
| **To be completed by the candidate by marking a cross (×).** | 1 | 2 | 3 | 4 | |
| To be completed by the examiners: | | | | | |
| | | | | | |
| | | | | | |

1) (a) Write down two common uses of database triggers.

[4 marks]

**ANSWER IN THIS BOX**

- Provide sophisticated auditing
- Prevent invalid transactions
- Enforce referential integrity (either those actions not supported by declarative constraints or across nodes in a distributed database)
- Enforce complex business rules
- Enforce complex security authorizations
- Provide transparent event logging
- Automatically generate derived column values
- Enable building complex views that are updatable
- Track database events

**Any two of the above.**

(b) Write down three differences between triggers and stored procedures.

[6 marks]

**ANSWER IN THIS BOX**

Triggers are implicitly called by database itself while Stored Procedure has to be manually called by user.

Stored Procedure can pass the parameters which is not a case with Triggers.

While creating a Trigger, triggering event and action have to be specified, which is not a case with Stored Procedure.

A Trigger can call the specific Stored Procedure in it but the reverse is not true.

**Any three of the above.**

(c) Consider the tables Employee and Evaluation as given below to keep track of employees, and their job evaluation records.

Employee (Employee_id, First_name, Job_id, Salary, Department_id);
Evaluation (Employee_id NUMBER, Evaluation_date DATE , Job_id CHAR(05)
Department_id NUMBER, Score NUMBER );

Write down a stored procedure ***Add_EmpEvaluation (Employee_id, Sysdate, Score)*** to insert records to the Evaluation table with respect to the evaluation of an employee. **Sysdate** is the date on which the evaluation is performed and **Score** is an integer value between 1 and 10, given based on the performance of the employee.

[7 marks]

**ANSWER IN THIS BOX**

```
create or replace PROCEDURE Add_EmpEvaluation
        (emp_id IN  employee.employee_id%TYPE, today IN DATE, score IN
                                                    evaluation.score%TYPE)
 AS
  -- placeholders for variables
  job_id            employee.job_id%TYPE;
  department_id  employee.department_id%TYPE;

BEGIN
   -- extracting values from employees for later insertion into evaluations
   SELECT e.job_id, e.department_id INTO job_id, department_id FROM employee e
    WHERE e.employee_id = emp_id;

   -- inserting a new row of values into evaluations table
   INSERT INTO evaluation VALUES (
     emp_id,              -- employee_id
     today,               -- evaluation_date
     job_id,              -- job_id
     department_id,       -- department_id
     score);              -- total_score

COMMIT;
END add_eval;
```

(d) Write down a trigger named ***Salary_Increment*** to increase salary of an employee as given below (based on the score value) upon insertion of his/her evaluation record into the evaluation table.

- 20% salary increment if score is >= 8
- 10% salary increment if score >= 6 and score < 8

[8 marks]

**ANSWER IN THIS BOX**

```
CREATE OR REPLACE TRIGGER Salary_Increment
 AFTER INSERT ON evaluations

 -- REFERENCING NEW AS new_rec

  FOR EACH ROW
   BEGIN

     if (:new.score >= 8)then
    UPDATE performance
     SET salary = salary * 1.2
      WHERE employee_id =:new.employee_id;
      END IF;

     if (:new.score >= 6 and :new.score < 8 )then
      UPDATE performance
      SET salary = salary * 1.1
        WHERE employee_id =:new.employee_id;
      END IF;

    END;
```

2) Consider the database represented below with respect to a reputed international company. The **Works_On** relation keeps track of which employee is working on which project. Each relation has its keys underlined.

```
Employee(EmpId,Ename, Designation, Salary, Gender, Dept_Code)
Works_On(EmpId, ProjId, Hours)
Project(ProjId, Pname, Budget, Duration, StartDate, Funded_Org)
```

- Suppose that the Employee file has $r_e$ = 30,000 records with blocking factor $bfr_e$ = 10 records/block.
- There are 150 distinct Dept_Code values. It is assumed that all the employees are evenly distributed among the departments.
- There are 100 projects. Many employees may not work for any of the projects and hence assume that there may be about 250 employees per project.

(a) Consider the following query

```
SELECT  E.name, E.Salary, E.Designation
FROM  Employee E
WHERE E.Dept_Code = 'SALE';
```

Suppose a multi level index on the nonkey attribute **Dept_Code** on Employee file is to be created and each index entry is 16 bytes long with block size B = 512 bytes. Calculate the total number of block accesses required to perform the query above if

i) Clustered multi level index on Dept_Code is to be used.
ii) Secondary index on Dept_Code is to be used.

[7 marks]

**ANSWER IN THIS BOX**

$Bfr_1$ of the index file → [512 / 16] (0.5 marks)

= 32 entries / block

Clustered Index

- Number of index entries is equal to the number of clusters in the data file, which is 150. (0.5 marks)
- The number of first level blocks = [150 / $Bfr_1$] (0.5 marks)

= [150 / 32]

= 5 Blocks

- The number of secondary level blocks = [5 / 32] (0.5 marks)

1 Block

There are two levels for clustered index. (0.5 marks)

Secondary Index

- Number of index entries is equal to the number of records in the data file which is 30,000 (0.5 marks)

- The number of first level blocks

b1 = [30000 / 32] = 938 blocks (0.5 marks)

b2 = [932 / 32] = 30 blocks (0.5 marks)

b3 = [30 / 32] = 1 block (0.5 marks)

There are three (3) levels in secondary index. (0.5 marks)

Number of records per dep-code = [30000 / 150] = 200 (0.5 marks)
Number of blocks per dep-code = [200 / 10] = 20 blocks (0.5 marks)
Block accesses through clustered index = 2+20 = 22 blocks (0.5 marks)
Block accesses through secondary index = 3+200 = 203 blocks (0.5 marks)

(b) Consider the following query:

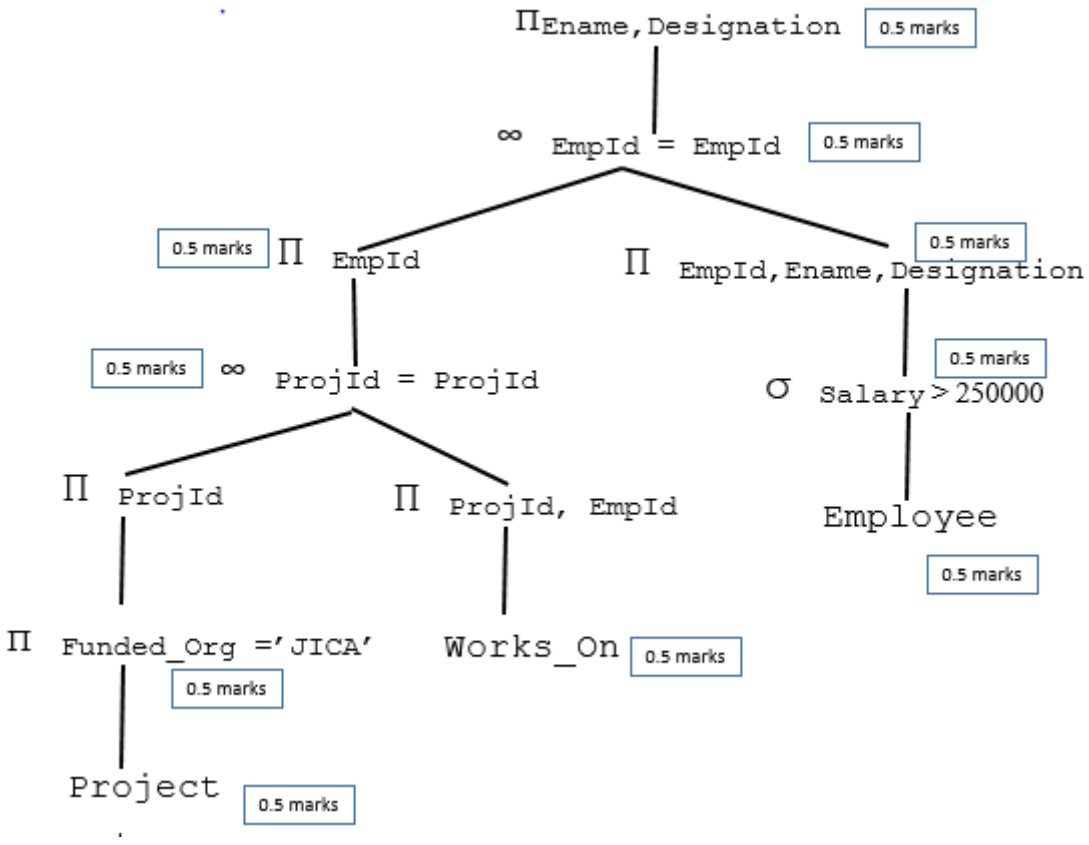
```
SELECT E.Ename, E.Designation
 FROM Employee E, Works_On W, Project P
   WHERE E.EmpId=W.EmpId AND W.ProjId = P.ProjId
       AND E.Salary > 250000 AND P.Funded_Org = 'JICA';
```

Suppose that there are about 5 projects funded by JICA and about 10% of the employees are earning more than Rs.250,000/=.

(i) Draw the optimized query tree for the query given above.

[5 marks]

**ANSWER IN THIS BOX**



(ii) Write down the corresponding relational algebra expression for the query tree that you have produced in b(i) above.

[5 marks]

**ANSWER IN THIS BOX**

(1 mark for each correct relational algebra expression)

$R1 = \pi_{ProjId}(\sigma_{Funded\_Org='JICA'}(Project))$

$R2 = \pi_{ProjId, EmpId}(Works\_On)$

$R3 = R1 \infty_{ProjId = ProjId} R2$

$R4 = \pi_{EmpId,Ename,Designation}(\sigma_{Salary > 25000}(Employee))$

$R5 = \pi_{Ename, Designation}(R3 \infty R4)$

(iii) Based upon the optimal query tree produced in b(i) above, determine the indexing mechanisms of the tables suggesting index-only plans where necessary.

[4 marks]

**ANSWER IN THIS BOX**

B+ tree/Hash Index on composite attributes (Funded_org, ProjId) on Project – Index Only Plan for Project file.
B+ tree/Hash index on ( ProjId, Empid) on Works_On - index only plan for Works_On file.
B+ tree index on Salary on Employee.

**Marks will be allocated accordingly for alternative answers.**

( c) Assume that the following query is frequently used to find the project ID relevant to any given employee name.

```
Select W.ProjId
 From Works_On W
 Where Exists (Select * From Employee E
                          Where E.EmpId = W.EmpId
                          And   E.Ename = 'Kasun Dias');
```

Re-write the query to improve the performance. Briefly explain how the re-written query will improve the performance. It is not necessary to write indexing for the query.

[4 Marks]

**ANSWER IN THIS BOX**

```
Select W.ProjId
      From Works_On W, Employee E
         Where E.EmpId = W.EmpId
                  And   E.Ename = 'Kasun Dias';
```

In correlated subqueries inner query cannot be evaluated independently of the outer query because the inner subquery uses the values of the parent statement. That is, the inner subquery is performed for each row in the outer query. That is, results of the subquery are dependent upon the row which is being evaluated with respect to the parent statement. Making the query un-correlated and un-nested will enable the query optimizer to decide upon the best possible plan without getting constrained to explicit separation of inner and outer queries.

3) (a) Explain the difference between a serial schedule and a serializable schedule using an appropriate example.

[2 Marks]

**ANSWER IN THIS BOX**

A schedule is serial if the operation of the various transactions are executed in sequence.
e.g. S1 : r0(x) r0(y) w0(x) r1(y) r1(x) w1(y) r2(x) r2(y) r2(z) w2(z)

A schedule is serializable if it is equivalent to one of the serial schedules with the same Transactions.
e.g. S1 : r0(x) r1(y) r3(x) r1(x) w1(y) r1(y) r2(y) r2(z) w1(x) w2(z)

(b) Consider the following schedule created using six transactions. Is this schedule conflict-serializable? Give reasons. If the above schedule is serializable, determine an equivalent serial schedule.

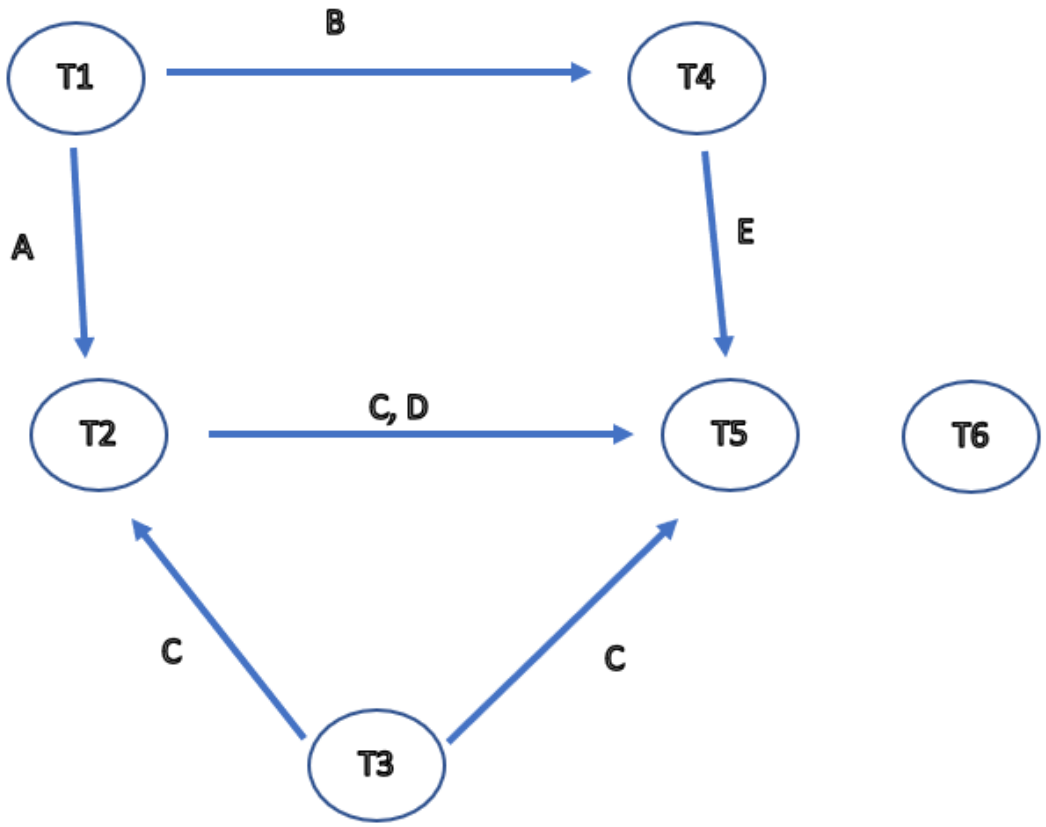
S1: $W_1(A)$, $R_2(A)$, $W_1(B)$, $R_1(F)$, $R_6(G)$, $W_3(C)$, $R_2(C)$, $R_4(B)$, $W_2(D)$, $R_4(F)$, $W_4(E)$, $W_5(C)$, $R_5(D)$, $R_2(D)$, $W_5(E)$, $R_6(F)$, $R_1(B)$, $W_6(G)$

**Note:** $R_1(X)$ denotes Transaction 1 Read X value.
$W_2(X)$ denotes Transaction 2 Write X value.

[10 Marks]

**ANSWER IN THIS BOX**

The precedence graph contains no cycles; therefore, the schedule is conflict serializable. (2 marks)
Correct Precedence graph (6 marks)




Any correct Serial Schedule – (2 marks)

**Ex: T1, T3, T2, T4, T5, T6**

(c) Consider the concurrent execution of two transactions T1 and T2 as given below.

| T1 | T2 |
|---|---|
| READ(A) | |
| | READ(A) |
| A=A-1 | |
| WRITE(A) | |
| | A=A+10 |
| | WRITE(A) |
| COMMIT | |
| | COMMIT |

Assume that the concurrency control is done using a time stamping protocol and all the timestamps are initialized to 0. The timestamp-based scheduler will allocate timestamps for transactions using a counter starting at 1. Clearly indicate the timestamps of T1, T2 and read, write timestamps of item X at each step until both T1 and T2 get committed.

[5 marks]

**ANSWER IN THIS BOX**

T1 = 0, T2 = 0 R_TS(X) = 0 , W_TS(X) = 0

R1(A) T1 = 1, T2 = 0 R_TS(X) = 1 , W_TS(X) = 0

R2(A) T1 = 1, T2 = 2 R_TS(X) = 2 , W_TS(X) = 0

A= A-1 T1 = 1, T2 = 2 R_TS(X) = 2 , W_TS(X) = 0

W1(A) Can not perform since R_TS(X) > TS(T1) ; roll back

A=A+10 T1 = 1, T2 = 2 R_TS(X) = 2 , W_TS(X) = 0

W2(A) T1 = 1, T2 = 2 R_TS(X) = 2 , W_TS(X) = 2

T1 restart with a new time stamp T1=3

R1(A) T1 = 3, T2 = 2 R_TS(X) = 3 , W_TS(X) = 2

A= A-1 T1 = 3, T2 = 2 R_TS(X) = 3 , W_TS(X) = 2

W1(A) T1 = 3, T2 = 2 R_TS(X) = 3 , W_TS(X) = 3

(0.5 marks for each step → 5 marks)

(d) Deferred update is a technique used for database recovery from non-catastrophic transaction failures.

i) Explain briefly, the reason for using No-UNDO/REDO algorithm in differed update.

[2 Marks]

**ANSWER IN THIS BOX**

The database is never updated until after the transaction commits, and there is never a need to UNDO any operations. Therefore, this technique is known as the NO-UNDO/REDO algorithm.

The REDO is needed in case the system fails after the transaction commits but before all its changes are recorded in the database. In this case, the transaction operations are redone using the log entries.

(ii) Consider the database log given below at the point of a system crash. Assuming that the deferred update protocol is used for recovery, identify what will happen to transactions T1 to T4, and what will be the values of the data items A, B, C, D, E, and F on disk after recovery?

The format of a log entry: <Transaction ID, Data Item, New value, Old value>

| | |
|---|---|
| 1 | <START T1> |
| 2 | <T1, A, 500, 100> |
| 3 | <T1, B, 200, 50> |
| 4 | <START T2> |
| 5 | <T1, A, 750, 500> |
| 6 | <T2, C, 80, 20> |
| 7 | <COMMIT T1> |
| 8 | <START T3> |
| 9 | <T3, C, 120, 80> |
| 10 | <T3, D, 50, 20> |
| 11 | <T2, E, 60, 40> |
| 12 | <CHECKPOINT> |
| 13 | <T2, C, 90, 120> |
| 14 | <COMMIT T2> |
| 15 | <START T4> |
| 16 | <T4, F, 30, 10> |
| 17 | <COMMIT T3> |
| 18 | <T4, E, 100, 60> |
| 19 | <T4, B, 140, 200> |
| | **[System Crash]** |

[6 Marks]

**ANSWER IN THIS BOX**

T1→ No need to redo since it has committed before the last check point.

T2, T3 → will be redone

T4 → is ignored since it has not reached until the commit point and no undo is required in differed update protocol

**A = 750**

**B=200**

**C=90**

**D=50**

**E=60**

**F=10**

4 (a) Briefly explain the correctness rules of database fragmentation.

[6 Marks]

**ANSWER IN THIS BOX**

**Completeness** - Decomposition of relation R into fragments R1, R2, . . . , Rn is complete iff each data item in R can also be found in some Ri.

**Reconstruction** - If relation R is decomposed into fragments R1, R2, . . . , Rn, then using union or join relational operators appropriately, R should be reconstructed from its fragments.

**Disjointness** - If relation R is decomposed into fragments R1, R2, . . . , Rn and data item di appears in fragment Rj , then di should not appear in any other fragment Rk, $k \neq j$ (exception: primary key attribute for vertical fragmentation)

(b) Consider the following relational schema where Emp_ID is the primary key of the Employee relation. Let E1, E2, E3 be fragmentations of Employee defined as follows. Are any of the correctness rules of fragmentation violated? Justify your answer.

**Employee (Emp_ID, Name, Designation, Salary, Dept_ID)**

$E1 = \pi_{Emp\_ID, Name, Salary}\ \sigma_{Salary \geq 150000}\ (Employee)$

$E2 = \sigma_{Salary \leq 150000}\ \pi_{Emp\_ID, Name, Salary}\ (Employee)$

$E3 = \pi_{Designation, Dept\_ID}\ (Employee)$

[6 Marks]

**ANSWER IN THIS BOX**

Reconstruction is violated because E3 does not contain the primary key, thus these three fragments (E1, E2, E3) could not be reconstructed into Employee, because there would be a lack of mapping.

Completeness would be fine, as all tuples and columns are stored.

Disjointness also violated – in E1 and E2, tuples where Salary=150000 appear to be duplicated.

(c) Given below is a set of relations available in a database of a company. The company has four (04) main departments namely HR, Finance, Marketing and Product Development. Under the Product Development department, there are two divisions for short-term projects and long-term projects. Assume that all the departments and divisions have their own database and maintain their data as a synchronized distributed database system.

**Employee (Emp_ID, E_Name, Designation, Joined_Date, Dept_Name, Salary)**

**Project (P_ID, P_Name, Budget, Client, Duration)**

The following rules are applicable when sharing data with departments.

- Each department has the rights to maintain data about their employees through replicating data in their sites except the salary information, which is handled only by the HR department.
- The Product development department has access to project information except for the project budget, which is only visible to the Finance department.
- The Short-term projects division handles projects where duration ≤ 1 year and long-term project division handles projects where duration > 1 year.

Assume that based on the above rules, a distributed database has to be designed. Give fragmented relations for each department and division. Express the fragmentation conditions using relational algebra for each fragment. Identify the type of fragmentation applied and indicate how you would reconstruct the original relation.

[13 Marks]

**ANSWER IN THIS BOX**

(1 mark each for the correct fragmentation written using relational algebra –> max 8 marks )

HR Department

Emp1 = $\pi_{Emp\_ID, Salary}$ (Employee)

Emp2 = $\pi_{Emp\_ID, E\_Name, Designation, Joined\_Date, Dept\_Name}$ (Employee)

Finance Department

Emp3 = $\sigma_{\text{Dept\_Name = "Finance"}}$ (Emp2)

Proj1 = $\pi_{\text{P\_ID, Budget}}$ (Project)

Proj2 = $\pi_{\text{P\_ID, P\_Name, Client, Duration}}$ (Project)

Marketing Department

Emp4 = $\sigma_{\text{Dept\_Name = "Marketing"}}$ (Emp2)

Product Development Department

Emp5 = $\sigma_{\text{Dept\_Name = "Product Development"}}$ (Emp2)

Proj3 = $\sigma_{\text{duration} \leq 1}$ (Proj2) optional

Proj4 = $\sigma_{\text{duration} > 1}$ (Proj2) optional

Short-Term Project Team

Proj3 = $\sigma_{\text{duration} \leq 1}$ (Proj2)

Long-Term Project Team

Proj4 = $\sigma_{\text{duration} > 1}$ (Proj2)

(1 mark each for identifying correct reconstruction method → 2 marks)

Proj3 $\cup$ Proj4 → Proj2

Proj1 $\infty$ Proj2→ Project

Emp3 $\cup$ Emp4 $\cup$ Emp5 → Emp2

Emp1 $\infty$ Emp2→ Employee

(3 marks for identifying fragmentation type)

Employee is vertically fragmented to Emp1 and Emp2

| |
|---|
| Emp2 is horizontally fragmented to Emp3, Emp4, Emp5 and resultant relations are mixed fragmented |
| Project is vertically fragmented to Proj1 and Proj 2 |
| Proj2 is horizontally fragmented to Proj3 and Proj4 based on duration ≤ 1 and duration > 1. Resultant are mixed fragmented. |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |

**********