Corporate Payroll Management System

Aim

To develop a Corporate Payroll Management System in C++ using data structures such as linked lists, queues, stacks, trees, graphs, and hash tables. The system manages employee records, processes salaries, handles corrections, represents departmental hierarchy, tracks interoffice relationships, and enables fast employee ID lookup.

Algorithm

1. Start the program.
2. Store employee details using a linked list for dynamic insertion and deletion.
3. Use a queue to process monthly salary payments in FIFO order.
4. Use a stack to manage salary corrections and rollback operations.
5. Represent the company department structure using a tree.
6. Represent interoffice connections using a graph.
7. Use a hash table for fast employee ID lookup.
8. Display employee details and salary processing information.
9. End the program.

Source Code

```
#include <iostream>
#include <list>
#include <queue>
#include <stack>
#include <unordered_map>
#include <vector>
using namespace std;
```

```
struct Employee {
   int id;
   string name;
   double salary;
};

list<Employee> employeeList;
queue<Employee> salaryQueue;

stack<Employee> correctionStack;

struct Department {
   string name;
   vector<Department*> subDepartments;
};

unordered_map<string, vector<string>> officeGraph;

unordered_map<int, Employee> employeeHash;


void addEmployee(int id, string name, double salary) {
   Employee e = {id, name, salary};
   employeeList.push_back(e);
   employeeHash[id] = e;
}

void enqueueSalary(Employee e) {
   salaryQueue.push(e);
}

void processSalary() {
   if (salaryQueue.empty()) {
      cout << "No salaries to process\n";
      return;
   }
   Employee e = salaryQueue.front();
   salaryQueue.pop();
   correctionStack.push(e);
   cout << "Salary processed for " << e.name
```

```
        << " Amount: " << e.salary << endl;
}

void rollbackSalary() {
    if (correctionStack.empty()) {
        cout << "No correction available\n";
        return;
    }
    Employee e = correctionStack.top();
    correctionStack.pop();
    cout << "Salary correction applied for " << e.name << endl;
}

void displayEmployees() {
    cout << "\nEmployee List:\n";
    for (auto e : employeeList)
        cout << "ID: " << e.id << " Name: " << e.name
             << " Salary: " << e.salary << endl;
}

int main() {

    addEmployee(101, "Ruthramurthi", 50000);
    addEmployee(102, "Arun", 45000);

    enqueueSalary(employeeHash[101]);
    enqueueSalary(employeeHash[102]);

    cout << "Salary Processing:\n";
    processSalary();
    processSalary();

    rollbackSalary();

    officeGraph["Chennai"].push_back("Bangalore");
    officeGraph["Bangalore"].push_back("Hyderabad");

    displayEmployees();

    cout << "\nHash Lookup (ID 101): "
```

```
        << employeeHash[101].name << endl;

    return 0;
}
```

Output

Salary Processing:

Salary processed for Ruthramurthi Amount: 50000

Salary processed for Arun Amount: 45000

Salary correction applied for Arun

Employee List:

ID: 101 Name: Ruthramurthi Salary: 50000

ID: 102 Name: Arun Salary: 45000

Hash Lookup (ID 101): Ruthramurthi