Assignment Instructions: Employee Management System Using Pointers, Classes, Virtual Functions, and Abstract Classes

Objective:

In this assignment, you will design an Employee Management System using C++ concepts such as pointers, inheritance, abstract classes, and virtual functions. The goal is to manage a list of full-time and part-time employees, calculate their salaries, and dynamically manage the employee data using pointers.

Requirements:

1. Abstract Class & Inheritance:

- Create an abstract base class `Employee` with the following:
  - Protected members: `name` (string) and `empID` (integer).
  - A pure virtual function `printDetails()` to print employee information.
  - A pure virtual function `calculateSalary()` to calculate and return the salary of the employee.

2. Derived Classes:

- Derive two classes from the `Employee` class:
  - FullTimeEmployee:
    - Private member: `basicSalary` (double).
    - Override the `printDetails()` function to display full-time employee details.
    - Override the `calculateSalary()` function to return the basic salary.
  - PartTimeEmployee:

- Private members: `hourlyWage` (double) and `hoursWorked` (integer).

- Override the `printDetails()` function to display part-time employee details.

- Override the `calculateSalary()` function to return the total salary (i.e., hourly wage × hours worked).

3. Pointers:

- Use pointers to dynamically create and manage instances of `Employee` objects.

- Store employee objects in an array of pointers (`Employee*`), and allow dynamic memory management using pointers.

4. User Input:

- The user will input details for multiple employees (number of employees is user-defined).

- For each employee, the program will ask for:

  - Name and employee ID.

  - Employee type (1 for Full-Time, 2 for Part-Time).

  - Basic salary (for full-time) or hourly wage and hours worked (for part-time).

5. Dynamic Memory Management:

- Use dynamic memory allocation (`new` and `delete`) to store employee objects.

- Ensure proper memory management by freeing dynamically allocated memory at the end of the program.

Program Structure:

1. Base Class: `Employee`

- Abstract class with pure virtual functions `printDetails()` and `calculateSalary()`.

2. Derived Classes:

- FullTimeEmployee: Inherits from `Employee`, implements salary calculation based on a fixed salary.

- PartTimeEmployee: Inherits from `Employee`, implements salary calculation based on hourly wages and hours worked.

3. Pointer Array:

- Use a pointer array to dynamically manage multiple employees.

- Allocate memory for each employee object based on user input and store the objects in the pointer array.

4. User Interaction:

- Implement functions to input employee details, display employee information, and calculate their salaries.

- Use the pointer array to traverse and manage employees dynamically.

Example Input:

```
Enter the number of employees: 3

Enter details for Employee 1:
Enter name: Sumanth
Enter employee ID: 123
Enter employee type (1 for Full-Time, 2 for Part-Time): 1
Enter basic salary: 80000

Enter details for Employee 2:
Enter name: Burugula
Enter employee ID: 111
Enter employee type (1 for Full-Time, 2 for Part-Time): 2
Enter hourly wage: 100
Enter hours worked: 40

Enter details for Employee 3:
Enter name: John
Enter employee ID: 112
Enter employee type (1 for Full-Time, 2 for Part-Time): 1
Enter basic salary: 100000
```

```
Employee Details and Salaries:
Full-Time Employee:
Name: Sumanth, ID: 123, Basic Salary: $80000
Calculated Salary: $80000

Part-Time Employee:
Name: Burugula, ID: 111, Hourly Wage: $100, Hours Worked: 40
Calculated Salary: $4000

Full-Time Employee:
Name: John, ID: 112, Basic Salary: $100000
Calculated Salary: $100000
```

```
Enter the number of employees: 2

Enter details for Employee 1:
Enter name: sumanth
Enter employee ID: 11
Enter employee type (1 for Full-Time, 2 for Part-Time): 4
Invalid employee type! Skipping this entry.

Enter details for Employee 1:
Enter name:
```

Deliverables:

1. Source Code: Submit your C++ code in a `.cpp` file.

2. Program Output: Include a screenshot of the output from your program after it has been successfully executed.

Grading Criteria:

- Correctness of code (40%)

- Proper use of inheritance and pointers, Abstract classes and Virtual functions (30%)

- Code readability, structure, and comments (15%)

- Output clarity (15%)

Good luck with your assignment!