```cpp
#include <iostream>
using namespace std;

int main() {
    const int MAX_EMPLOYEES = 10;
    const int NAME_LENGTH = 10;
    const int POSITION_LENGTH = 10;  // Changed from 10 to 20
    const int PHONE_LENGTH = 10;

    string names[MAX_EMPLOYEES];
    string positions[MAX_EMPLOYEES];
    string phoneNumbers[MAX_EMPLOYEES];
    int ids[MAX_EMPLOYEES];
    int ages[MAX_EMPLOYEES];
    double salaries[MAX_EMPLOYEES];
    int count = 0;

```

1. Variable Declarations
- Initializes constants and arrays to store employee data.
  - `MAX_EMPLOYEES = 10`: Maximum number of employees the system can handle.
  - `NAME_LENGTH = 10`: Maximum allowed characters for an employee's name.
  - `POSITION_LENGTH = 10` (later changed to 20): Maximum allowed characters for an employee's position.
  - `PHONE_LENGTH = 10`: Exact required digits for a phone number.
- Arrays:
  - `names`, `positions`, `phoneNumbers`: Store strings for employee details.
  - `ids`, `ages`, `salaries`: Store integers/doubles for employee ID, age, and salary.:
  - `count`: Tracks the current number of employees in the system.

```
18    while (true) {
19        cout << "\nEmployee Management System\n";
20        cout << "1. Add Employee\n";
21        cout << "2. View All Employees\n";
22        cout << "3. Search Employee\n";
23        cout << "4. Exit\n";
24        cout << "Enter your choice: ";
25
26        int choice;
27        cin >> choice;
28
29        if (choice == 1) {
30            if (count >= MAX_EMPLOYEES) {
31                cout << "Maximum employees reached!\n";
32                continue;
33            }
34
35            // Name input (max 10 characters)
36            cout << "Enter employee name (max 10 chars): ";
37            string name;
38            cin >> name;
39            if (name.length() > NAME_LENGTH) {
40                cout << "Name too long! Using first 10 characters.\n";
41                name = name.substr(0, NAME_LENGTH);
42            }
43            names[count] = name;
```

2. Main Menu Loop

- Displays a menu and handles user input for navigating the system.
  1. Add Employee: Allows adding a new employee (checks if maximum capacity is reached).
  2. View All Employees: Displays a list of all employees.
  3. Search Employee: (Code not fully shown) Likely searches for an employee by ID or name.
  4. Exit: Terminates the program.

- Input Handling:
  - Reads the user's choice (`cin >> choice`).
  - Uses `if-else` to execute the corresponding functionality.
  - Validates the name length (truncates to 10 characters if too long).

```
        // Position input (now max 20 characters)
        cout << "Enter employee position (max 10 chars): ";
        string position;
        cin >> position;
        if (position.length() > POSITION_LENGTH) {
            cout << "Position too long!\n";
            cin>>position;
        }
        positions[count] = position;

        cout << "Enter employee ID: ";
        cin >> ids[count];
```

3. Position and ID Input
- Collects and validates the employee's position and ID.
-
  - Prompts for a position (max 10 characters, but note the comment says 20).
  - Checks length; if too long, it prints an error .
  - Stores the position in the `positions` array.

- ID Input:
  - Directly reads the employee ID and stores it in the `ids` array.

```
        // Phone number input (exactly 10 digits)
        cout << "Enter employee phone number (10 digits): ";
        string phone;
        cin >> phone;
        while (phone.length() != PHONE_LENGTH) {
            cout << "Invalid length! Enter exactly 10 digits: ";
            cin >> phone;
        }
        phoneNumbers[count] = phone;

        cout << "Enter employee age: ";
        cin >> ages[count];

        cout << "Enter employee salary: ";
        cin >> salaries[count];

        count++;
        cout << "Employee added successfully!\n";
```

4. **Phone, Age, and Salary Input
- Collects and validates phone number, age, and salary.
- Phone Number Input:
  - Requires exactly 10 digits.
  - Uses a `while` loop to repeatedly prompt the user until a valid phone number is entered.

- Age and Salary Input:
  - Directly reads age and salary without validation (e.g., no checks for negative values).
- Completion:
  - Increments `count` to reflect the new employee.
  - Prints a success message.

```
        } else if (choice == 2) {
            if (count == 0) {
                cout << "No employees in the system.\n";
                continue;
            }

            cout << "\nEmployee List:\n";
            cout << "ID\tName\t\tPosition\t\t\tPhone\t\tAge\tSalary\n";
            cout << "-------------------------------------------------------------
            for (int i = 0; i < count; i++) {
                cout << ids[i] << "\t"
                     << names[i] << "\t";
                if (names[i].length() < 8) cout << "\t";
                cout << positions[i] << "\t";
                if (positions[i].length() < 16) cout << "\t";
                cout << phoneNumbers[i] << "\t"
                     << ages[i] << "\t$"
                     << salaries[i] << "\n";
            }

        } else if (choice == 3) {
            /* ... rest of the search code remains exactly the same ... */
        } else if (choice == 4) {
            cout << "Exiting the system. Goodbye!\n";
            break;
        } else {
            cout << "Invalid choice. Please try again.\n";
        }
    }

    return 0;
}
```

5. View All Employees and Exit

- Displays all employees or exits the system.
- View All Employees:
  - Checks if there are no employees (`count == 0`) and displays a message if so.
  - Prints a formatted table with columns: ID, Name, Position, Phone, Age, Salary.
  - Uses `\t` for alignment and adjusts spacing based on field lengths.

- Exit:
  - Prints a goodbye message and breaks the loop to exit the program.
- Invalid Choice:
  - Handles invalid menu inputs with an error message.

Summary of Functions:
1. Initialization: Sets up data structures for employee records.
2. Menu Loop: Provides navigation and input handling.
3. Add Employee:
   - Validates and stores name, position, phone, ID, age, and salary.
   - Enforces length constraints for name, position, and phone.
4. View Employees: Displays all records in a tabular format.
5. **Exit**: Cleanly terminates the program.