Problem Statement: **Dynamic Memory Management in a Mini Banking System**

You are tasked with implementing a simple banking system in C that uses dynamic memory management to handle multiple customer accounts. Each customer has the following details:

- **Name**: The customer's full name.
- **Account Number**: A unique identifier for the customer's account.
- **Account Balance**: The current balance in the customer's account.
- **Transaction History**: A list of transactions (deposits/withdrawals) performed by the customer, where each transaction includes a description.

Requirements:

1. **Dynamic Memory Allocation**: You must use pointers to dynamically allocate memory for customer details and transaction history.
2. **Customer Structure**: Design a `Customer` structure with the following fields:
    - `char* name`: A pointer to dynamically allocated memory for the customer's name.
    - `int account_number`: A unique integer representing the account number.
    - `float balance`: The account balance as a floating-point number.
    - `char** transaction_history`: A pointer to an array of strings that stores transaction descriptions.
    - `int transaction_count`: The number of transactions recorded for the customer.
3. **Functions to Implement**:
    - `create_customer()`: Dynamically allocates memory for a new customer, including the customer's name and an empty transaction history.
    - `update_balance()`: Updates the account balance (either deposit or withdrawal).
    - `add_transaction()`: Adds a transaction to the customer's transaction history.
    - `print_customer_info()`: Displays all the information about a customer, including the transaction history.
    - `free_customer()`: Frees all dynamically allocated memory for a customer.
4. **Transaction History Management**: The system should store transaction records dynamically. If the maximum allowed number of transactions is reached, no more transactions can be added for that customer.
5. **Memory Management**: Properly manage memory allocation and deallocation to avoid memory leaks.

Example Workflow:

- **Create Customer**: A user creates a new customer with their name, account number, and initial balance.
- **Update Balance**: The user deposits or withdraws money, updating the balance.
- **Transaction Recording**: Every time a deposit or withdrawal occurs, the system logs the transaction with a description.
- **Display Information**: The system can display the customer's details and their transaction history at any time.
- **Free Memory**: When a customer is no longer needed, all allocated memory should be freed.
