Banking System

Specifications:

- Variables: Account number, name, balance, and transaction type.
- Static & Const: Static variable for total accounts; const for maximum transaction limits.
- Switch Case: Menu for creating accounts, deposits, withdrawals, and balance inquiry.
- Looping Statements: Loop to process transactions.
- Pointers: Pointer for updating account balances.
- Functions: Separate functions for each banking operation.
- Arrays: Store account details.
- Structures: Structure for account details.
- Nested Structures: Nested structures for account and transaction details.
- Unions: Union for different transaction types.
- Nested Unions: Nested union for handling various financial instruments.
- Output Expectations: Display account details and transactions.

Menu Example:

1. Create Account
2. Deposit Money
3. Withdraw Money
4. Check Balance
5. Exit

Code:

```
#include <stdio.h>
#include <string.h>
```

```
#define MAX_ACCOUNTS 100
#define MAX_TRANSACTIONS 1000
#define MAX_DEPOSIT 100000
#define MAX_WITHDRAW 50000

// Structure for transaction details
struct Transaction
{
   union
   {
      struct
      {
         float depositAmount;
         float withdrawAmount;
      } types;
   } transactionType;
   char type; // D-deposit and W-withdraw
};

// Structure for account details
struct Account
{
   int accountNumber;
   char name [50];
   float balance;
   struct Transaction transactions [MAX_TRANSACTIONS];
   int transactionCount;
```

```
};

// Array to store account details
struct Account accounts [MAX_ACCOUNTS];
static int totalAccounts = 0; // Static variable for total accounts

// Function declarations
void createAccount();
void depositMoney();
void withdrawMoney();
void checkBalance();
void updateBalance(struct Account *account, float amount, char type);

int main()
{
   int option;
   do
   {
      printf("\nBanking System Menu\n");
      printf("1. Create Account\n");
      printf("2. Deposit Money\n");
      printf("3. Withdraw Money\n");
      printf("4. Check Balance\n");
      printf("5. Exit\n");
      printf("Enter the option: ");
      scanf("%d", &option);
      switch (option)
```

```
        {
            case 1: createAccount();
                    break;
            case 2: depositMoney();
                    break;
            case 3: withdrawMoney();
                    break;
            case 4: checkBalance();
                    break;
            case 5: printf("Exiting the Banking System\n");
                    break;
            default:printf("Invalid option\n");
        }
    } while (option != 5);
    return 0;
}

// Function to create an account
void createAccount()
{
    if (totalAccounts >= MAX_ACCOUNTS)
    {
        printf("Maximum account limit reached\n");
        return;
    }
    struct Account *newAccount = &accounts[totalAccounts];
    newAccount->accountNumber = totalAccounts + 1;
```

```
    printf("Enter name for Account %d: ", newAccount->accountNumber);
    scanf(" %[^\n]", newAccount->name);
    newAccount->balance = 0.0;
    newAccount->transactionCount = 0;
    printf("Account created successfully! Account Number: %d\n", newAccount->accountNumber);
    totalAccounts++;
}

// Function to deposit money
void depositMoney()
{
    int accNo;
    float amount;
    printf("Enter Account Number: ");
    scanf("%d", &accNo);
    if (accNo < 1 || accNo > totalAccounts)
    {
        printf("Invalid Account Number\n");
        return;
    }
    struct Account *account = &accounts[accNo - 1];
    printf("Enter amount to deposit: ");
    scanf("%f", &amount);
    if (amount <= 0 || amount > MAX_DEPOSIT)
    {
        printf("Invalid amount! Maximum deposit limit is %.2f\n", (float)MAX_DEPOSIT);
```

```
        return;
    }
    updateBalance(account, amount, 'D');
    printf("Deposit successful! Current Balance: %.2f\n", account->balance);
}

// Function to withdraw money
void withdrawMoney()
{
    int accNo;
    float amount;
    printf("Enter Account Number: ");
    scanf("%d", &accNo);
    if (accNo < 1 || accNo > totalAccounts)
    {
        printf("Invalid Account Number\n");
        return;
    }
    struct Account *account = &accounts [accNo - 1];
    printf("Enter amount to withdraw: ");
    scanf("%f", &amount);

    if (amount <= 0 || amount > MAX_WITHDRAW || amount > account->balance)
    {
        printf("Invalid amount! Maximum withdraw limit is %.2f or insufficient balance\n", (float)MAX_WITHDRAW);
        return;
```

```
        }
        updateBalance(account, -amount, 'W');
        printf("Withdrawal successful! Current Balance: %.2f\n", account->balance);
    }

    // Function to check balance
    void checkBalance()
    {
        int accNo;
        printf("Enter Account Number: ");
        scanf("%d", &accNo);
        if (accNo < 1 || accNo > totalAccounts)
        {
            printf("Invalid Account Number\n");
            return;
        }
        struct Account *account = &accounts [accNo - 1];
        printf("Account number: %d\n", account->accountNumber);
        printf("Name: %s\n", account->name);
        printf("Balance: %.2f\n", account->balance);
        printf("Transactions -   \n");
        for (int i = 0; i < account->transactionCount; i++)
        {
            if (account->transactions[i].type == 'D')
                printf("Deposit:%.2f\n",account->transactions[i].
                        transactionType.types.depositAmount);
            else if (account->transactions[i].type == 'W')
```

```
                printf("Withdrawal:%.2f\n",account->transactions[i].
                        transactionType.types.withdrawAmount);
        }
    }

    // Function to update account balance
    void updateBalance(struct Account *account, float amount, char type)
    {
        account->balance += amount;
        struct Transaction *newTransaction = &account->transactions[account-
    >transactionCount++];
        newTransaction->type = type;
        if (type == 'D')
            newTransaction->transactionType.types.depositAmount = amount;
        else if (type == 'W')
            newTransaction->transactionType.types.withdrawAmount = -amount;
    }
```

Output:

```
Banking System Menu
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Check Balance
5. Exit
Enter the option: 1
Enter name for Account 1: Nanditha M
Account created successfully! Account Number: 1
```

Banking System Menu

1. Create Account

2. Deposit Money

3. Withdraw Money

4. Check Balance

5. Exit

Enter the option: 1

Enter name for Account 2: Monisha M

Account created successfully! Account Number: 2

Banking System Menu

1. Create Account

2. Deposit Money

3. Withdraw Money

4. Check Balance

5. Exit

Enter the option: 2

Enter Account Number: 1

Enter amount to deposit: 10000

Deposit successful! Current Balance: 10000.00

Banking System Menu

1. Create Account

2. Deposit Money

3. Withdraw Money

4. Check Balance

5. Exit

Enter the option: 2

Enter Account Number: 2

Enter amount to deposit: 8000

Deposit successful! Current Balance: 8000.00

Banking System Menu

1. Create Account

2. Deposit Money

3. Withdraw Money

4. Check Balance

5. Exit

Enter the option: 3

Enter Account Number: 1

Enter amount to withdraw: 5000

Withdrawal successful! Current Balance: 5000.00

Banking System Menu

1. Create Account

2. Deposit Money

3. Withdraw Money

4. Check Balance

5. Exit

Enter the option: 2

Enter Account Number: 2

Enter amount to deposit: 2000

Deposit successful! Current Balance: 10000.00

Banking System Menu

1. Create Account

2. Deposit Money

3. Withdraw Money

4. Check Balance

5. Exit

Enter the option: 4

Enter Account Number: 2

Account number: 2

Name: Monisha M

Balance: 10000.00

Transactions -

Deposit: 8000.00

Deposit: 2000.00

Banking System Menu

1. Create Account

2. Deposit Money

3. Withdraw Money

4. Check Balance

5. Exit

Enter the option: 4

Enter Account Number: 1

Account number: 1

Name: Nanditha M

Balance: 5000.00

Transactions -

Deposit: 10000.00

Withdrawal: 5000.00

Banking System Menu

1. Create Account

2. Deposit Money

3. Withdraw Money

4. Check Balance

5. Exit

Enter the option: 5

Exiting the Banking System