DSA MINI PROJECT

UE22CS252A

3rd Semester, Academic Year 2023

BANK MANAGEMENT SYSTEM

Team Members :

M C Krishna Kumar - PES2UG22CS281

Lohit Kumar Nagarur - PES2UG22CS280

MADDINALA VENKAT CHARAN - PES2UG22CS289

M SAI NITHIN - PES2UG22CS284

SYNOPSIS

Our program is for a basic banking system that allows users to perform various operations, including creating accounts, depositing money, checking account balances, transferring money between accounts, withdrawing money, and deleting accounts. Data structure used is Doubly Linked List

C Code Function Explanations

```
int acct_number()
```

Generates a random 6-digit account number.

```
acct* check(acct *head, int key)
```

Searches for an account with a given account number in the linked list.

```
void insert(acct** root, acct* temp)
```

Inserts a new account node into a binary search tree based on the account number.

```
acct* create_acct(acct* root)
```

Creates a new bank account by taking user input for account holder name and generates a unique account number. Inserts the account into the binary search tree.

`void deposit(acct* root, int acct_num, float money)`

Deposits a specified amount into the account with the given account number.

`void check_balance(acct* root, int acct_num)`

Checks and prints the balance of the account with the given account number.

`void transfer(acct* root, int acct_num1, int acct_num2, float money)`

Transfers a specified amount from one account to another account.

`void withdraw(acct* root, int acct_num, float money)`

Withdraws a specified amount from the account with the given account number.

`void delete_account(acct* root, int acct_num)`

Deletes the account with the given account number from the binary search tree.

`int main()`

The main function that provides a menu for users to interact with different banking operations, such as creating an account, depositing money, checking balance, transferring money, withdrawing money, deleting an account, and exiting the program.

This program simulates a basic banking system using a DLL to manage customer accounts and provides various banking operations for account management.

Code :

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

typedef struct ACCOUNT
{
    int account_number;
    char acct_holder_name[50];
    float balance;
    struct ACCOUNT* left;
    struct ACCOUNT* right;
} acct;
```

```
int acct_number()
{
    srand(time(NULL)); // Seed the random number generator only once at the
beginning
  int acctnum = rand() % 900000 + 100000;
    return acctnum;
}

acct* check(acct* root, int acct_num)
{
    if (root == NULL)
        return NULL;
    else
    {
        acct* cur = root;
        while (cur != NULL && cur->account_number != acct_num)
            cur = (acct_num < cur->account_number) ? cur->left : cur->right;
        return cur;
    }
}


void insert(acct** root, acct* temp)
{
    acct* new_acc = (acct*)malloc(sizeof(acct));
    new_acc->account_number = temp->account_number;
    strcpy(new_acc->acct_holder_name, temp->acct_holder_name);  // Copy the
account holder name
    new_acc->balance = temp->balance;
    new_acc->left = NULL;
    new_acc->right = NULL;

    if (*root == NULL)
    {
        *root = new_acc;
    }
    else
    {
        acct* cur = *root;
        while (cur != NULL)
        {
            if (temp->account_number < cur->account_number)
            {
                if (cur->left == NULL)
                {
                    cur->left = new_acc;
                    new_acc->right = cur;
                    break;
                }
                cur = cur->left;
            }
            else
            {
                if (cur->right == NULL)
```

```
                {
                    cur->right = new_acc;
                    new_acc->left = cur;
                    break;
                }
                cur = cur->right;
            }
        }
    }
}

acct* create_acct(acct* root)
{
    acct* temp = (acct*)malloc(sizeof(acct));
    if (temp == NULL)
    {
        printf("Please try to create the account again\n");
        return root;
    }

    temp->balance = 0.0;
    temp->left = NULL;
    temp->right = NULL;

    printf("Please Enter Your Name: ");
    scanf("%s", temp->acct_holder_name);

    // Generate a unique account number
    temp->account_number = acct_number();

    // Use the corrected insert function to maintain order
    insert(&root, temp);

    printf("Thank you for creating an account in the bank.\n");
    printf("Name: %s\nAccount number: %d\nAccount has been created\n", temp-
>acct_holder_name, temp->account_number);

    return root;
}




void deposit(acct* root, int acct_num, float money)
{
    acct* cur = check(root, acct_num);
    if (cur != NULL)
    {
        cur->balance += money;
        printf("Successfully added %.2f amount in the account number %d\n", money,
acct_num);
    }
    else
    {
```

```
        printf("Could not find the bank account. Please try again later.\n");
    }
}

void check_balance(acct* root, int acct_num)
{
    acct* cur = check(root, acct_num);
    if (cur != NULL)
    {
        printf("Account holder name: %s\nAccount number: %d\nBalance: %.2f\n",
cur->acct_holder_name, cur->account_number, cur->balance);
    }
    else
    {
        printf("Could not find the account. Please try again.\n");
    }
}

void transfer(acct* root, int acct_num1, int acct_num2, float money)
{
    if (root == NULL)
    {
        printf("Could not find the account. Please try again later.\n");
        return;
    }

    if (acct_num1 == acct_num2)
    {
        printf("Self transfer\n");
        deposit(root, acct_num1, money);
        return;
    }

    acct* cur1 = check(root, acct_num1);
    acct* cur2 = check(root, acct_num2);

    if (cur1 == NULL || cur2 == NULL)
    {
        printf("Cannot make the transaction. Please try again with proper
information.\n");
        return;
    }
    else
    {
        cur1->balance -= money;
        cur2->balance += money;
        printf("Successfully transferred %.2f money from account number %d to
account number %d\n", money, acct_num1, acct_num2);
    }
}

void withdraw(acct* root, int acct_num, float money)
{
    acct* cur = check(root, acct_num);
```

```
    if (cur != NULL)
    {
        if (cur->balance >= money)
        {
            cur->balance -= money;
            printf("Successfully withdrew %.2f amount from the account number
%d\n", money, acct_num);
        }
        else
        {
            printf("Insufficient balance for withdrawal.\n");
        }
    }
    else
    {
        printf("Could not find the bank account. Please try again later.\n");
    }
}
void delete_account(acct** root, int acct_num)
{
    acct* cur = *root;
    acct* prev = NULL;

    while (cur != NULL && cur->account_number != acct_num)
    {
        prev = cur;
        cur = cur->right;
    }

    if (cur != NULL && cur->account_number == acct_num)
    {
        if (prev == NULL)
        {
            *root = cur->right;
        }
        else
        {
           prev->right = cur->right;
        }
        free(cur);
        printf("Account number %d has been deleted.\n", acct_num);
    }
    else
    {
        printf("Could not find the account. Please try again.\n");
    }
}
void writeToFile(acct* root, const char* filename){
    FILE* file = fopen(filename, "w");
    if (file == NULL)
    {
        printf("Error opening file for writing.\n");
        return;
    }
```

```
    acct* cur = root;
    fprintf(file, "Account Number,Holder Name,Balance\n\n");
    while (cur != NULL)
    {
        fprintf(file, "%d\t%s\t%.2f\n", cur->account_number, cur-
>acct_holder_name, cur->balance);
        cur = cur->right;
    }
    fclose(file);
    printf("Accounts have been written to %s\n", filename);
}

void empl(acct* root, const char *code) {
    if (strcmp(code, "222") == 0) {
        acct* current = root;
        writeToFile(root,"1.txt");
    } else {
        printf("You are not an employee!\n");
    }
}



int main()
{

    int user_acct_num = 0;
    int user_acct_num1 = 0, user_acct_num2 = 0;
    float amt = 0;
    char code[100];
    acct* root = NULL;

    while (1)
    {
        int ch;
        printf("\tWelcome to our Bank\nWe request you to choose one of the
following functions\n");
        printf("1. Create\n2. Deposit\n3. Check balance\n4. Transfer from one
account to another account\n5. Withdraw money\n6. Delete account\n7. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &ch);

        switch (ch)
        {
        case 1:
            root = create_acct(root);
            break;
        case 2:
            printf("Enter your account number: ");
            scanf("%d", &user_acct_num);
            printf("Enter the amount to deposit in the account: ");
            scanf("%f", &amt);
            deposit(root, user_acct_num, amt);
            break;
```

```
            case 3:
                printf("Enter your account number: ");
                scanf("%d", &user_acct_num);
                check_balance(root, user_acct_num);
                break;
            case 4:
                printf("Enter your account number: ");
                scanf("%d", &user_acct_num1);
                printf("Enter receiver's account number: ");
                scanf("%d", &user_acct_num2);
                printf("Enter the amount to be transferred: ");
                scanf("%f", &amt);
                transfer(root, user_acct_num1, user_acct_num2, amt);
                break;
            case 5:
                printf("Enter your account number: ");
                scanf("%d", &user_acct_num);
                printf("Enter the amount to withdraw: ");
                scanf("%f", &amt);
                withdraw(root, user_acct_num, amt);
                break;
            case 6:
                printf("Enter your account number: ");
                scanf("%d", &user_acct_num);
                delete_account(&root, user_acct_num);
                break;
            case 7:
                printf("Thank you for visiting our bank. Have a nice day.\n");
                exit(0);
            case 8:printf("Enter employee code: ");
            scanf("%s",code); // Remove the newline character
                empl(root, code);

                break;
            default:
                printf("Please try again.\n");
                continue;
            }
        }
        return 0;
    }
```