TABLE OF CONTENTS

ABSTRACT

This C program simulates a simple banking system that allows multiple users to perform basic banking operations. It employs a singly linked list to manage user accounts, each represented by a node containing the user's ID, name, password, account balance, and a pointer to the next node.

The program provides the following core functionalities:

1).Account Creation – Allows users to create a new bank account with a name, password (8-10 characters), and an initial deposit (Rs.3000–25000).

2).Deposit Money – Enables users to add funds to their account after successful password authentication.

3).Withdraw Money – Allows users to withdraw money, provided the amount does not exceed their balance.

4).Check Balance – Lets users view their current account balance upon authentication.

5).Money Transfer – Permits users to transfer funds to another valid account, with balance validation.

6).Account Deletion – Users can delete their accounts by providing correct credentials, and the balance is "returned" as part of the output.

The program emphasizes password-based access control and input validation to simulate basic banking security and functionality. It operates via a menu-driven interface and maintains accounts dynamically using dynamic memory allocation (malloc), thus avoiding fixed array limitations.

INTRODUCTION

Banking systems play a vital role in managing financial transactions and customer information securely and efficiently. This C program is a simplified model of a banking management system that provides essential banking operations such as account creation, deposit, withdrawal, balance inquiry, fund transfer, and account deletion.

The program uses a singly linked list data structure to dynamically manage user accounts, where each account is represented as a node containing relevant details like account ID, name, password, and balance. This allows the system to handle an arbitrary number of users efficiently without relying on fixed-size data structures.

A menu-driven interface guides the user through various operations. For each transaction, the system verifies user credentials (ID and password) to maintain a basic level of security. Input validations, such as password length checks and balance constraints, ensure logical consistency and mimic real-world banking rules.

This project demonstrates the practical use of fundamental concepts in C programming such as structures, pointers, dynamic memory allocation, and string manipulation, while simulating the core functionalities of a banking system.

SOURCE CODE

1. Start.

2. Display the main menu with options:

3. Take the user's choice as input.

4. Based on the selected option, perform the corresponding operation:

    - Create account
    - Deposit
    - Withdraw
    - Check Balance
    - Transfer Money
    - Delete Account
5. Repeat the menu until the user selects Exit.

6. Stop.

CODE

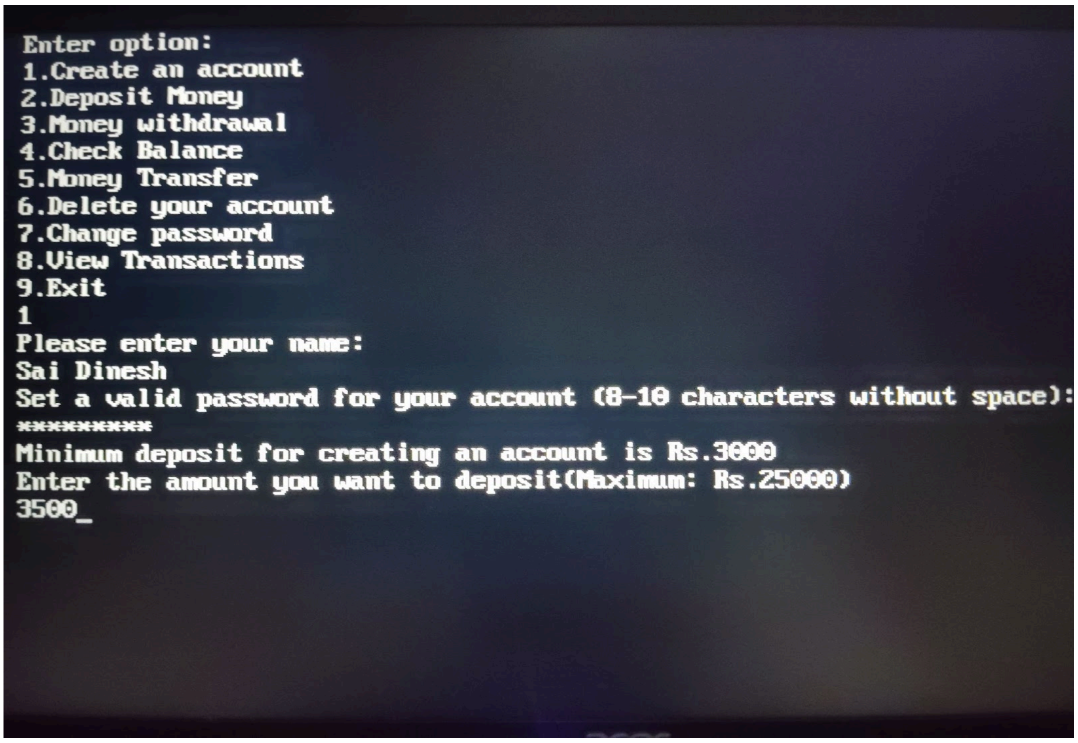
```
#include<ctype.h>
#include<conio.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct node {
   int balance;
   int id;
   char password[100];
   char name[100];
   char transactions[5][100];
   int t_index;
   struct node *link;
} *root;
int count = 0;
void update_file() {
   int i;
   FILE *fp = fopen("accounts.txt", "w");
   struct node *t = root;
   while (t != NULL) {
         fprintf(fp, "%d %s %s %d %d\n", t->id, t->name, t->password, t->balance,
t->t_index);
         for (i = 0; i < 5; i++) {
            fprintf(fp, "%s\n", t->transactions[i]);
         }
         t = t->link;
   }
   fclose(fp);
}
void create() {
   struct node *nn, *temp;
   int i,bal, valid = 0;
```

```
char na[100], pass[20], ch;
printf("Please enter your name:\n");
fflush(stdin);
gets(na);
do {
    i = 0;
    printf("Set a valid password for your account (8-10 characters without space):\n");
    while ((ch = getch()) != '\r' && i < 10) {
      if (ch == ' ') continue;
      pass[i++] = ch;
      printf("*");
    }
    pass[i] = '\0';
    printf("\n");

    if (strlen(pass) >= 8 && strlen(pass) <= 10)
      valid = 1;
    else
      printf("Invalid password length. Try again.\n");
} while (!valid);
printf("Minimum deposit for creating an account is Rs.3000\n");
printf("Enter the amount you want to deposit(Maximum: Rs.25000)\n");
scanf("%d", &bal);
if (bal < 3000 || bal > 25000) {
    printf("Invalid deposit amount. Please try again.\n");
    return;
}
nn = (struct node *)malloc(sizeof(struct node));
nn->link = NULL;
nn->balance = bal;
count++;
nn->id = count;
strcpy(nn->name, na);
strcpy(nn->password, pass);
```

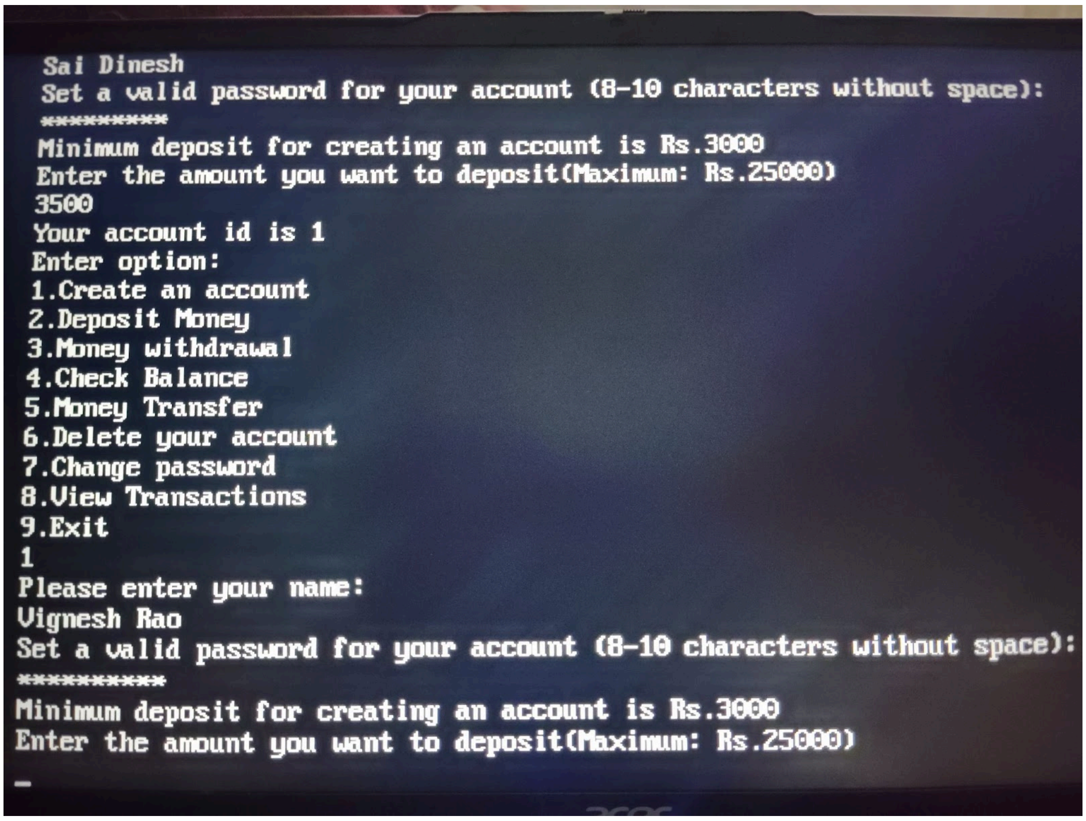
```
    nn->t_index = 0;
    for (i = 0; i < 5; i++)
        strcpy(nn->transactions[i], "");
    sprintf(nn->transactions[nn->t_index], "Account created with deposit Rs. %d", bal);
    nn->t_index = (nn->t_index + 1) % 5;
    temp = root;
    if (root == NULL)
        root = nn;
    else {
        while (temp->link != NULL) {
            temp = temp->link;
        }
        temp->link = nn;
    }
    printf("Your account id is %d\n", count);
    update_file();
}
void del_acc() {
    struct node *t = root, *p;
    int accid, op;
    char pass[20];
    printf("Do you want to delete your account?\n1.Yes\n2.No\n");
    scanf("%d", &op);
    if (op == 2)
        return;
    else {
        printf("Enter the account id:\n");
        scanf("%d", &accid);
        printf("Enter the password\n");
        fflush(stdin);
        scanf("%s", pass);
        while (t != NULL && t->id != accid) {
            p = t;
            t = t->link;
        }
        if (t == root && (strcmp(t->password, pass) == 0)) {
            p = t->link;
            printf("Please collect your balance amount at counter 1: Rs.%d\n", t->balance);
            free(t);
            root = p;
        }
        else if (t == NULL) {
            printf("Incorrect id..! Please try again..!\n");
            return;
        }
         else if (strcmp(t->password, pass) == 0) {
            p->link = t->link;
            printf("Name: %s", t->name);
            printf("Please collect your balance amount at counter 1: Rs.%d\n", t->balance);
            free(t);
            count--;
        }
        else
            printf("Invalid password\n");
    }
    update_file();
}

void withdraw() {
    int accid, amount;
    char pass[20];
    struct node *t = root;
    printf("Enter the acc id:\n");
    scanf("%d", &accid);
    printf("Enter the password\n");
    fflush(stdin);
    scanf("%s", pass);
    while (t != NULL && t->id != accid) {
        t = t->link;
    }
    if (t == NULL) {
        printf("Incorrect id..! Please try again..!\n");
        return;
    } else if (strcmp(t->password, pass) == 0) {
        printf("Enter the amount you want to withdraw\n");
        scanf("%d", &amount);
        if (amount > t->balance) {
            printf("Insufficient balance\n");
            return;
        } else {
            t->balance -= amount;
            printf("Please collect your money at counter 2: %d\nAccount balance: %d\n",
amount, t->balance);
            sprintf(t->transactions[t->t_index], "Withdrew Rs. %d", amount);
            t->t_index = (t->t_index + 1) % 5;
        }
    }
        else {
        printf("Invalid password\n");
    }
    update_file();
}
void transfer() {
    int accid, amount, id2;
    char pass[20];
    struct node *t = root, *p = root;
    printf("Enter your acc id:");
    scanf("%d", &accid);
    printf("Enter the password:");
    fflush(stdin);
    scanf("%s", pass);
    while (t != NULL && t->id != accid) {
        t = t->link;
    }
    if (t == NULL) {
        printf("Incorrect id..! Please try again..!\n");
        return;
    }
         else if (strcmp(t->password, pass) == 0) {
        printf("Enter receiver acc id:\n");
        scanf("%d", &id2);
        while (p != NULL && p->id != id2) {
            p = p->link;
        }
        if (p == NULL) {
            printf("Invalid id\n");
        }
        else {
            printf("Enter the amount you want to transfer:");
            scanf("%d", &amount);
            if (amount > t->balance) {
                printf("Insufficient balance\n");
                return;
            }
        else {
                t->balance -= amount;
                p->balance += amount;
                printf("Amount transferred successfully..!");
                printf("Account balance: %d\n", t->balance);
                sprintf(t->transactions[t->t_index], "Transferred Rs. %d to ID %d", amount,
p->id);
                t->t_index = (t->t_index + 1) % 5;
                sprintf(p->transactions[p->t_index], "Received Rs. %d from ID %d", amount,
t->id);
                p->t_index = (p->t_index + 1) % 5;
            }
            }
    }
             else {
            printf("Invalid password\n");
    }
    update_file();
}

void cb() {
    int accid;
    char pass[20];
    struct node *t = root;
    printf("Enter the acc id:\n");
    scanf("%d", &accid);
    printf("Enter the password\n");
    fflush(stdin);
    scanf("%s", pass);
    while (t != NULL && t->id != accid) {
            t = t->link;
    }
    if (t == NULL) {
            printf("Incorrect id..! Please try again..!\n");
            return;
    }
            else if (strcmp(t->password, pass) == 0) {
            printf("Account Balance:\n%d\n", t->balance);
    }
             else {
            printf("Invalid password\n");
    }
}
void deposit() {
    int accid, amount;
    char pass[20];
    struct node *t = root;
    printf("Enter the acc id:\n");
    scanf("%d", &accid);
    printf("Enter the password\n");
    fflush(stdin);
    scanf("%s", pass);
    while (t != NULL && t->id != accid) {
        t = t->link;
    }
    if (t == NULL) {
        printf("Incorrect id..! Please try again..!\n");
        return;
    }
        else if (strcmp(t->password, pass) == 0) {
        printf("Enter the amount you want to deposit:");
        scanf("%d", &amount);
        t->balance += amount;
        printf("Deposit successful\n");
        printf("Account Balance:\n%d\n", t->balance);
        sprintf(t->transactions[t->t_index], "Deposited Rs. %d", amount);
        t->t_index = (t->t_index + 1) % 5;
    }
        else {
        printf("Invalid password\n");
    }
    update_file();
}
void changepass() {
    struct node *t = root;
    int x;
    char s[100], pass[100];
    int found = 0;
    printf("Enter the ID to change password\n");
    scanf("%d", &x);
    printf("Enter the old password\n");
    scanf("%s", s);
    while (t != NULL) {
            if (t->id == x) {
                if (strcmp(t->password, s) == 0) {
                        found = 1;
                        break;
                }
            else {
                        printf("Incorrect password.\n");
                        return;
                }
            }
            t = t->link;
    }
    if (!found) {
            printf("Invalid ID.\n");
            return;
    }
    do {
            printf("\nSet a valid new password for your account (8-10 characters, no spaces):\n");
            scanf("%s", pass);
    } while (strlen(pass) < 8 || strlen(pass) > 10 || strchr(pass, ' '));
    strcpy(t->password, pass);
    printf("Password changed successfully.\n");
    update_file();
}
void load_from_file() {
    int i;
    size_t len;
    FILE *fp = fopen("accounts.txt", "r");
    struct node *nn, *last = NULL;
    if (!fp) return;
    while (1) {
        nn = (struct node *)malloc(sizeof(struct node));
        if (fscanf(fp, "%d %s %s %d %d", &nn->id, nn->name, nn->password,
&nn->balance, &nn->t_index) == 5) {
            for (i = 0; i < 5; i++) {
                fgets(nn->transactions[i], 100, fp);
                len = strlen(nn->transactions[i]);
                if (len > 0 && nn->transactions[i][len - 1] == '\n')
                    nn->transactions[i][len - 1] = '\0';
            }
            nn->link = NULL;
            if (root == NULL)
                root = nn;
            else
                last->link = nn;
            last = nn;
            if (nn->id > count)
                count = nn->id;
        }
        else {
            free(nn);
            break;
        }
    }
    fclose(fp);
}
void view_transactions() {
    int accid, i;
    char pass[20];
    struct node *t = root;
    int printed = 0;
    printf("Enter the account ID:\n");
    scanf("%d", &accid);
    printf("Enter the password:\n");
    fflush(stdin);
    scanf("%s", pass);
    while (t != NULL && t->id != accid) {
            t = t->link;
    }
    if (t == NULL) {
            printf("Incorrect account ID! Please try again.\n");
            return;
    }
    if (strcmp(t->password, pass) != 0) {
            printf("Invalid password!\n");
            return;
    }
    printf("Transaction history for %s (ID: %d):\n", t->name, t->id);
    for (i = 0; i < 5; i++) {
            int index = (t->t_index + i) % 5;
            if (strlen(t->transactions[index]) > 0) {
                printf("%d. %s\n", printed + 1, t->transactions[index]);
                printed++;
            }
    }
    if (printed == 0) {
            printf("No transactions found.\n");
    }
}
void main() {
    int op;
    clrscr();
    load_from_file();
    do {
            printf("Enter option:\n1.Create an account\n2.Deposit Money\n3.Money
withdrawal\n4.Check Balance\n5.Money Transfer\n6.Delete your account\n7.Change
password\n8.View Transactions\n9.Exit\n");
            scanf("%d", &op);
            switch (op) {
```

```
            case 1: create();
                    break;
            case 2: deposit();
                    break;
            case 3: withdraw();
                    break;
            case 4: cb();
                    break;
            case 5: transfer();
                    break;
            case 6: del_acc();
                    break;
            case 7: changepass();
                    break;
            case 8: view_transactions();
                    break;
        }
    } while (op != 9);
}
```

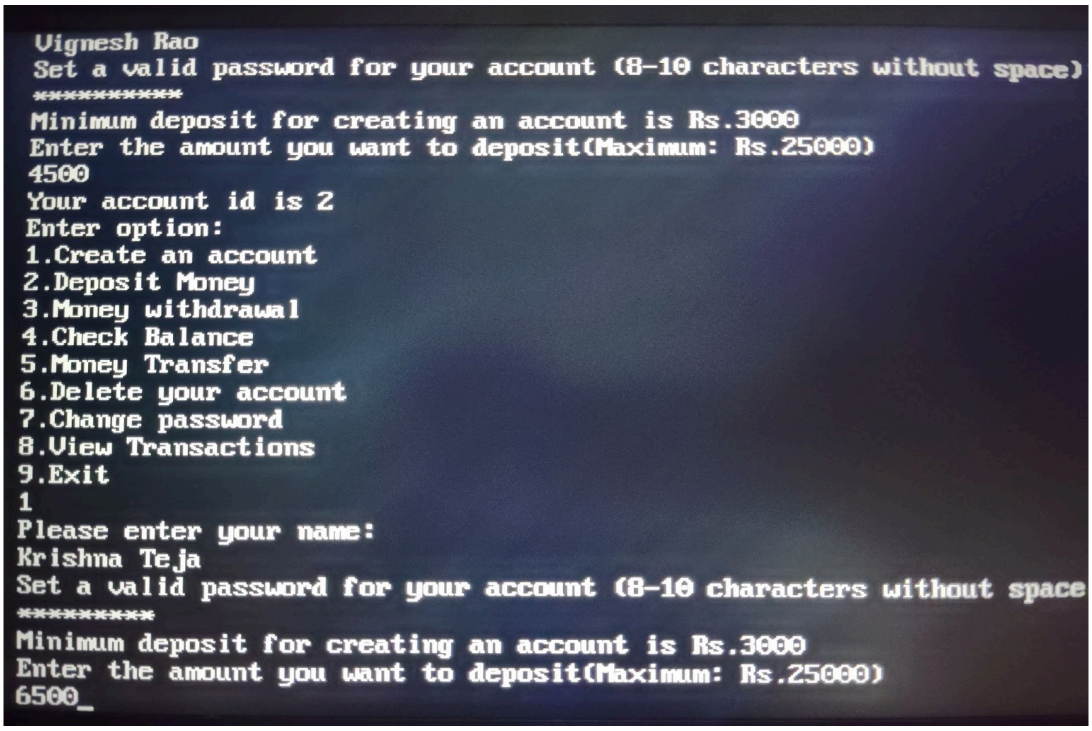
OUTPUT



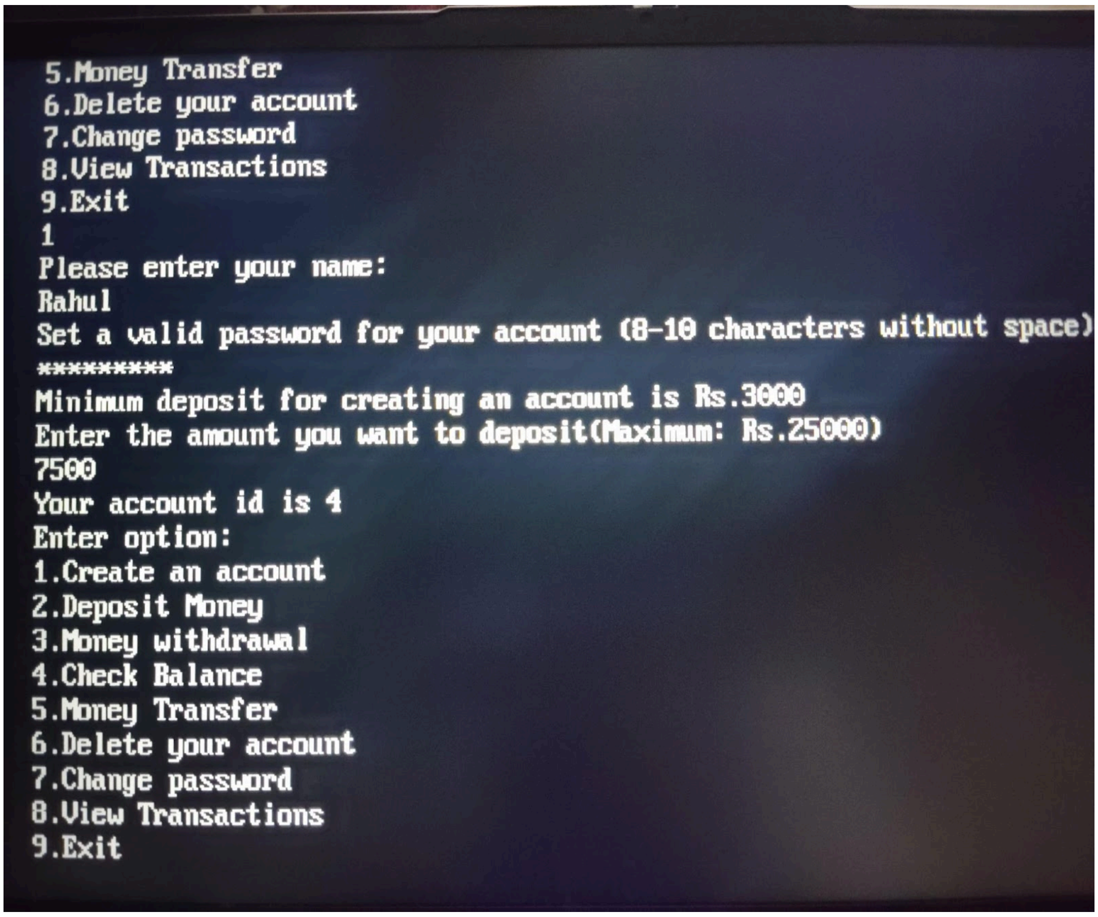
```
Enter option:
1.Create an account
2.Deposit Money
3.Money withdrawal
4.Check Balance
5.Money Transfer
6.Delete your account
7.Change password
8.View Transactions
9.Exit
1
Please enter your name:
Sai Dinesh
Set a valid password for your account (8-10 characters without space):
*********
Minimum deposit for creating an account is Rs.3000
Enter the amount you want to deposit(Maximum: Rs.25000)
3500_
```



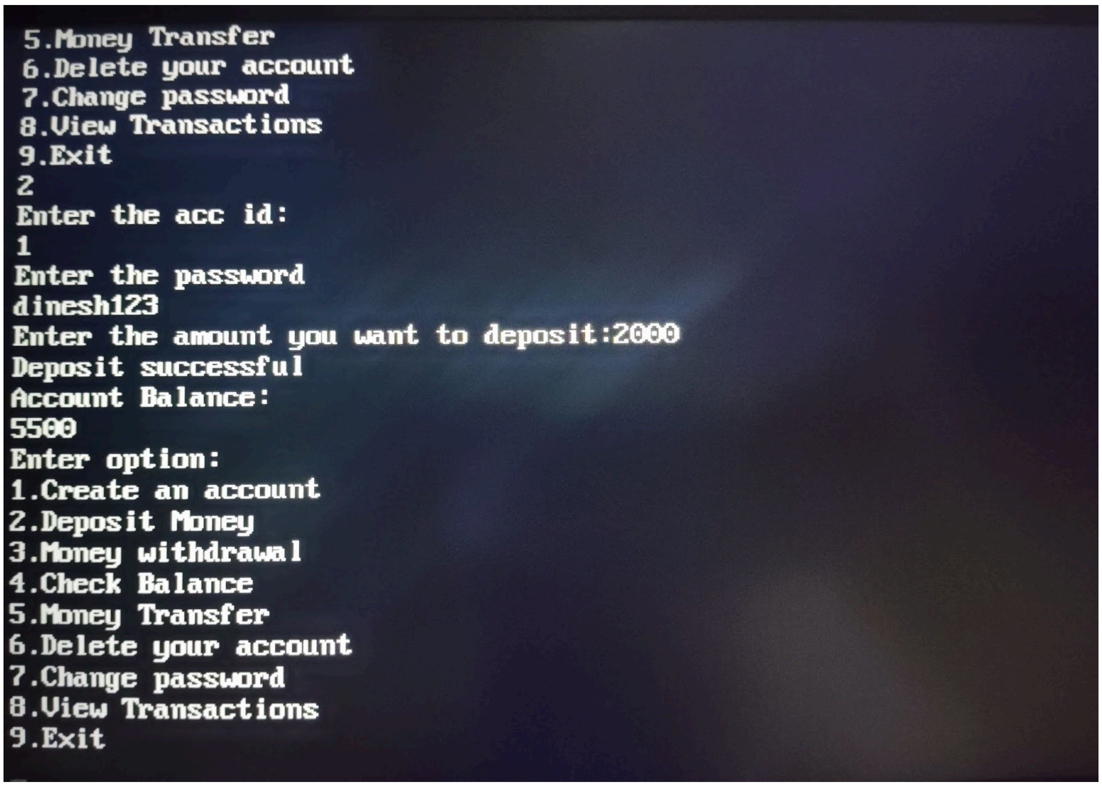
```
Sai Dinesh
Set a valid password for your account (8-10 characters without space):
**********
Minimum deposit for creating an account is Rs.3000
Enter the amount you want to deposit(Maximum: Rs.25000)
3500
Your account id is 1
Enter option:
1.Create an account
2.Deposit Money
3.Money withdrawal
4.Check Balance
5.Money Transfer
6.Delete your account
7.Change password
8.View Transactions
9.Exit
1
Please enter your name:
Vignesh Rao
Set a valid password for your account (8-10 characters without space):
**********
Minimum deposit for creating an account is Rs.3000
Enter the amount you want to deposit(Maximum: Rs.25000)
_
```



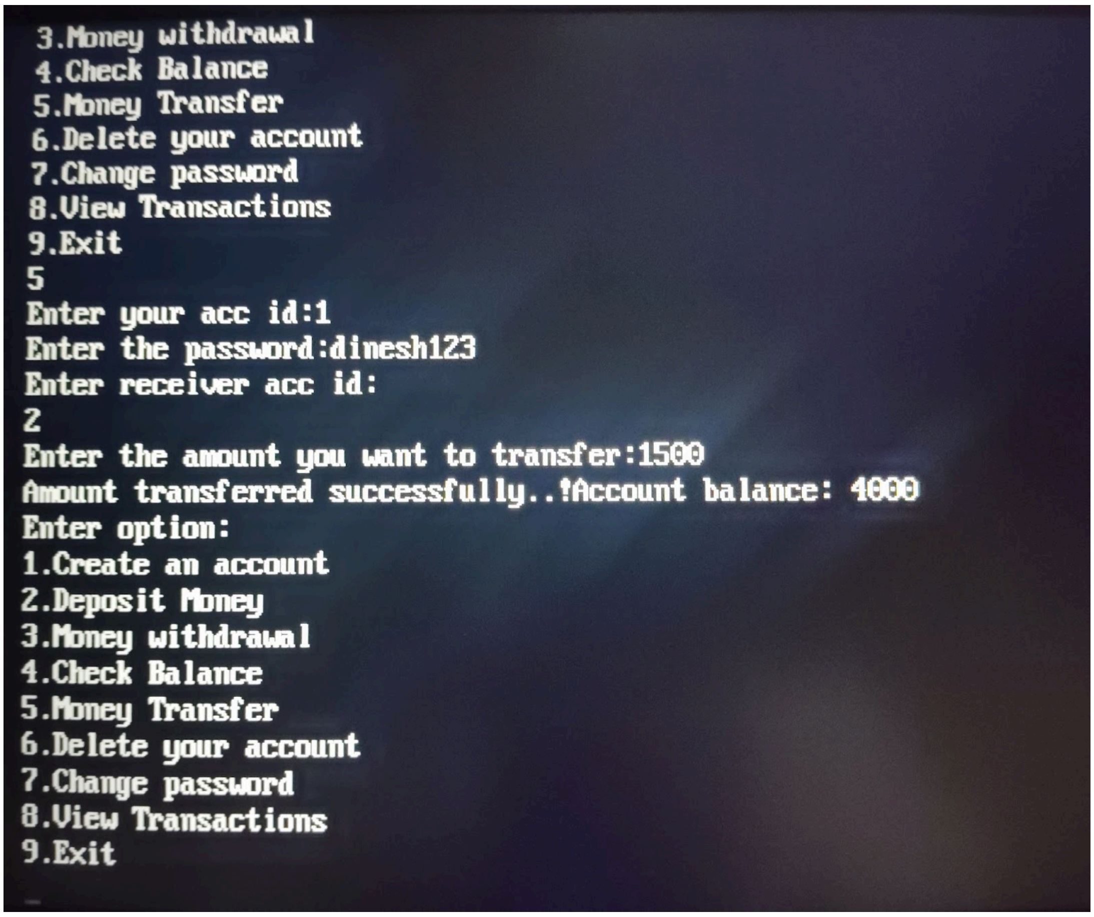
```
Vignesh Rao
Set a valid password for your account (8-10 characters without space)
***********
Minimum deposit for creating an account is Rs.3000
Enter the amount you want to deposit(Maximum: Rs.25000)
4500
Your account id is 2
Enter option:
1.Create an account
2.Deposit Money
3.Money withdrawal
4.Check Balance
5.Money Transfer
6.Delete your account
7.Change password
8.View Transactions
9.Exit
1
Please enter your name:
Krishna Teja
Set a valid password for your account (8-10 characters without space
**********
Minimum deposit for creating an account is Rs.3000
Enter the amount you want to deposit(Maximum: Rs.25000)
6500_
```



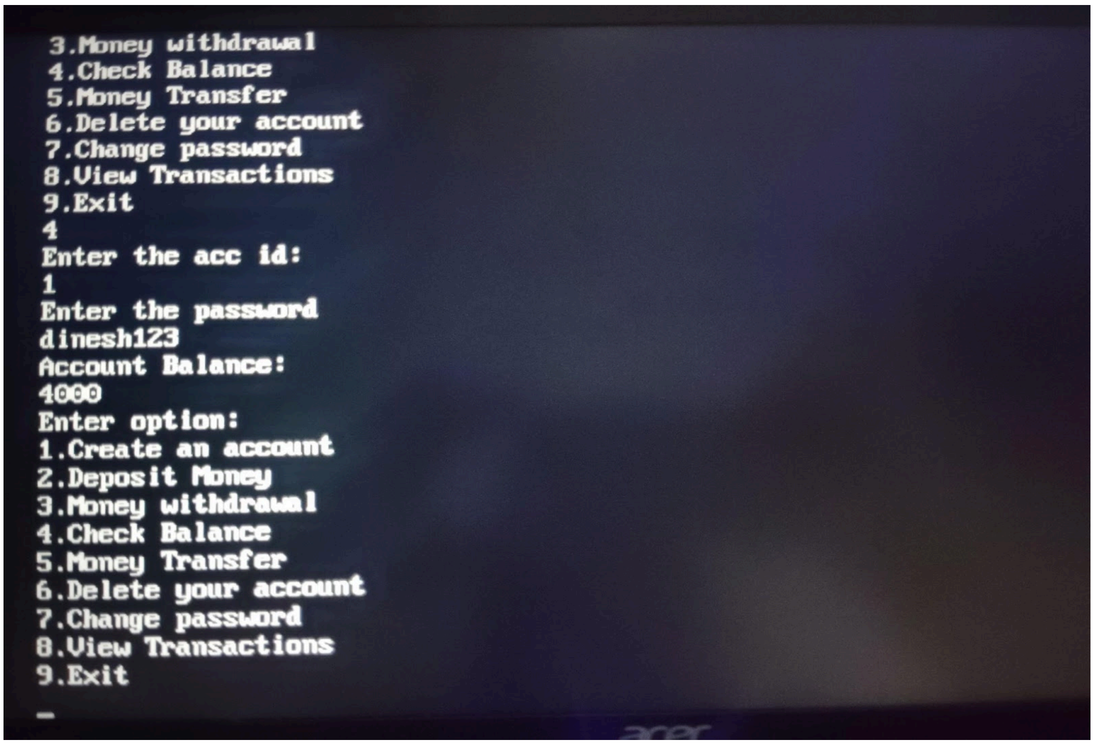
```
5.Money Transfer
6.Delete your account
7.Change password
8.View Transactions
9.Exit
1
Please enter your name:
Rahul
Set a valid password for your account (8-10 characters without space)
*********
Minimum deposit for creating an account is Rs.3000
Enter the amount you want to deposit(Maximum: Rs.25000)
7500
Your account id is 4
Enter option:
1.Create an account
2.Deposit Money
3.Money withdrawal
4.Check Balance
5.Money Transfer
6.Delete your account
7.Change password
8.View Transactions
9.Exit
```



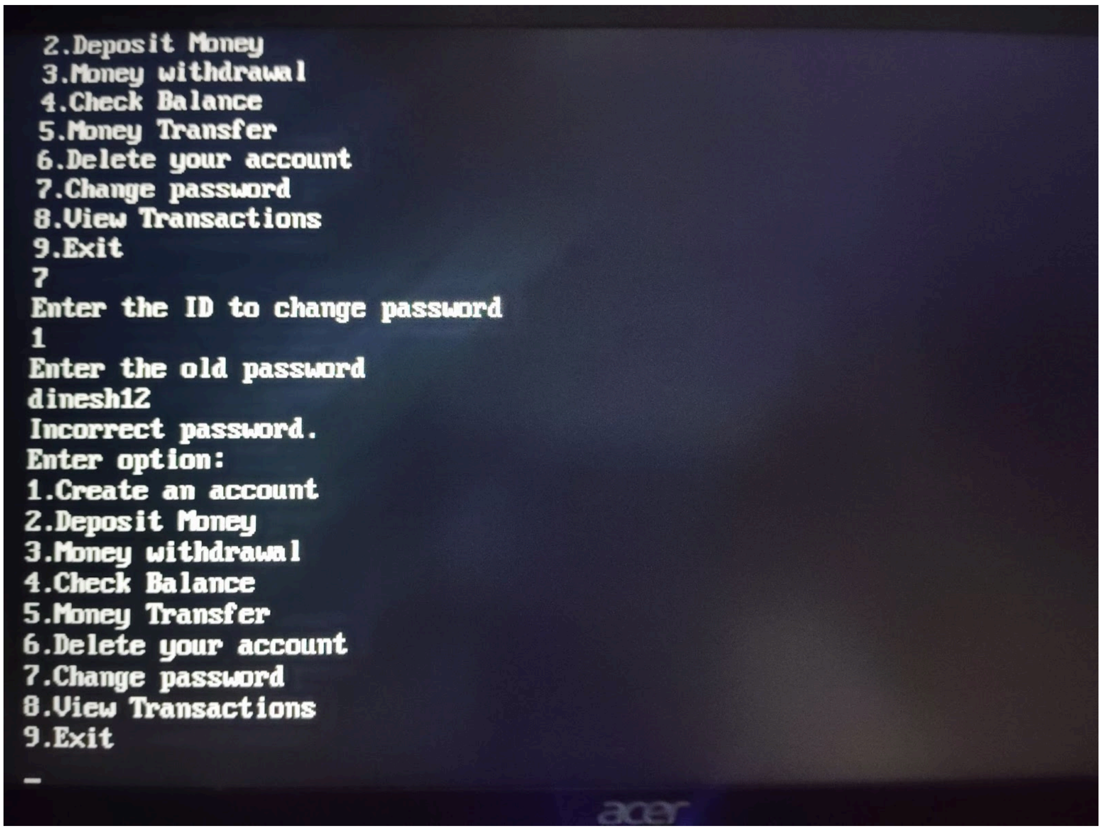
```
5.Money Transfer
6.Delete your account
7.Change password
8.View Transactions
9.Exit
2
Enter the acc id:
1
Enter the password
dinesh123
Enter the amount you want to deposit:2000
Deposit successful
Account Balance:
5500
Enter option:
1.Create an account
2.Deposit Money
3.Money withdrawal
4.Check Balance
5.Money Transfer
6.Delete your account
7.Change password
8.View Transactions
9.Exit
```



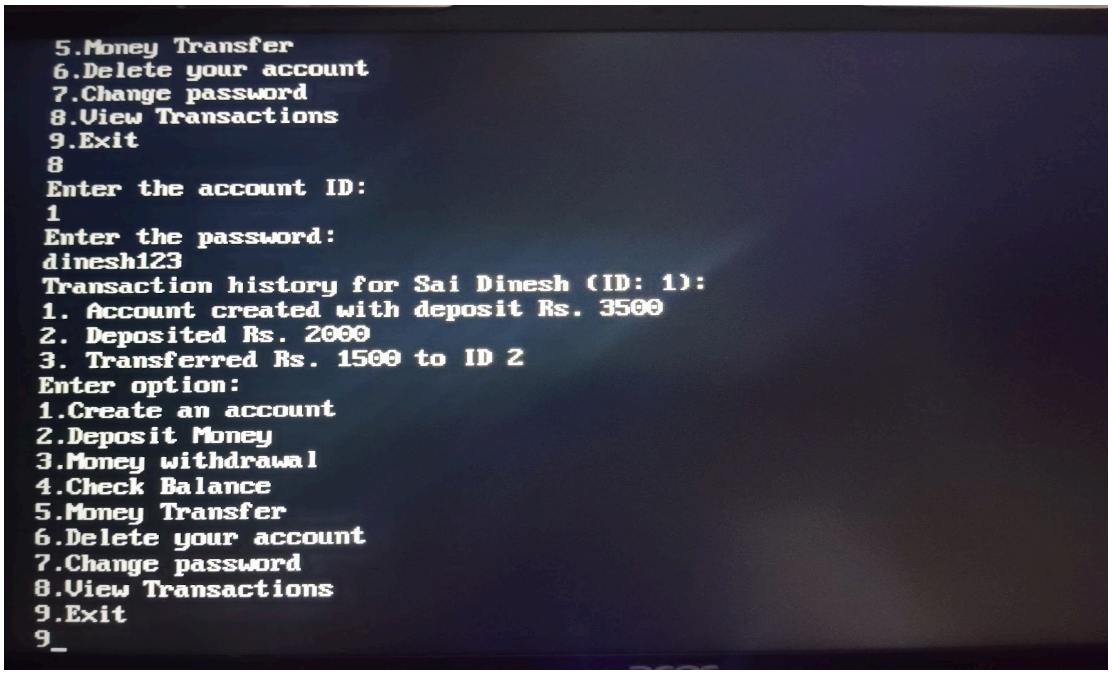
```
3.Money withdrawal
4.Check Balance
5.Money Transfer
6.Delete your account
7.Change password
8.View Transactions
9.Exit
5
Enter your acc id:1
Enter the password:dinesh123
Enter receiver acc id:
2
Enter the amount you want to transfer:1500
Amount transferred successfully..!Account balance: 4000
Enter option:
1.Create an account
2.Deposit Money
3.Money withdrawal
4.Check Balance
5.Money Transfer
6.Delete your account
7.Change password
8.View Transactions
9.Exit
```



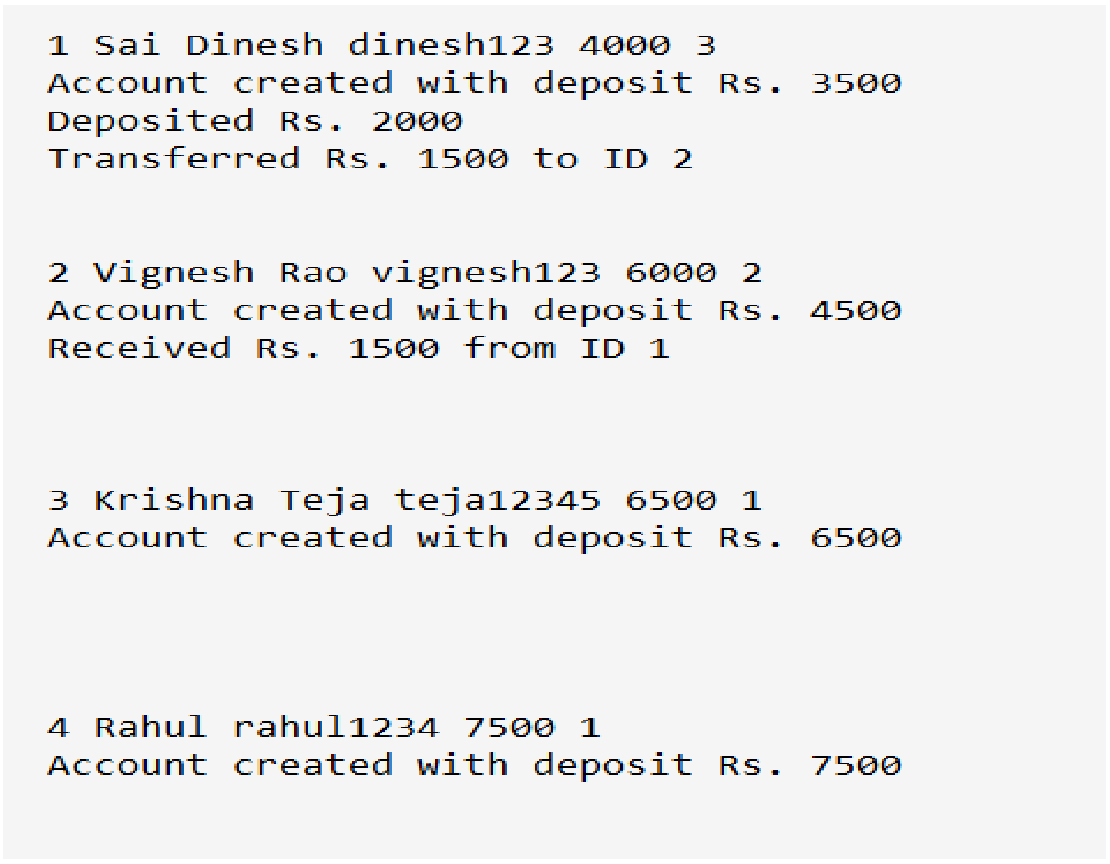
```
3.Money withdrawal
4.Check Balance
5.Money Transfer
6.Delete your account
7.Change password
8.View Transactions
9.Exit
4
Enter the acc id:
1
Enter the password
dinesh123
Account Balance:
4000
Enter option:
1.Create an account
2.Deposit Money
3.Money withdrawal
4.Check Balance
5.Money Transfer
6.Delete your account
7.Change password
8.View Transactions
9.Exit
_
```



```
2.Deposit Money
3.Money withdrawal
4.Check Balance
5.Money Transfer
6.Delete your account
7.Change password
8.View Transactions
9.Exit
7
Enter the ID to change password
1
Enter the old password
dinesh12
Incorrect password.
Enter option:
1.Create an account
2.Deposit Money
3.Money withdrawal
4.Check Balance
5.Money Transfer
6.Delete your account
7.Change password
8.View Transactions
9.Exit
_
```



```
5.Money Transfer
6.Delete your account
7.Change password
8.View Transactions
9.Exit
8
Enter the account ID:
1
Enter the password:
dinesh123
Transaction history for Sai Dinesh (ID: 1):
1. Account created with deposit Rs. 3500
2. Deposited Rs. 2000
3. Transferred Rs. 1500 to ID 2
Enter option:
1.Create an account
2.Deposit Money
3.Money withdrawal
4.Check Balance
5.Money Transfer
6.Delete your account
7.Change password
8.View Transactions
9.Exit
9_
```



```
1 Sai Dinesh dinesh123 4000 3
Account created with deposit Rs. 3500
Deposited Rs. 2000
Transferred Rs. 1500 to ID 2


2 Vignesh Rao vignesh123 6000 2
Account created with deposit Rs. 4500
Received Rs. 1500 from ID 1



3 Krishna Teja teja12345 6500 1
Account created with deposit Rs. 6500




4 Rahul rahul1234 7500 1
Account created with deposit Rs. 7500
```

CONCLUSION

In conclusion, this banking system program developed in C demonstrates a simple yet effective approach to handling basic banking operations such as account creation, deposits, withdrawals, balance inquiries, money transfers, and account deletion. By using structures and linked lists, it efficiently manages dynamic user data during runtime, allowing the program to handle multiple accounts without a fixed limit. The inclusion of password protection ensures that each transaction is secure and user-specific, simulating real-world authentication practices.

Although the system currently operates in-memory without persistent storage, it effectively illustrates key programming concepts such as pointer manipulation, dynamic memory allocation, and modular function design. These concepts are critical for developing larger and more complex software systems. Furthermore, the program demonstrates how to build a menu-driven interface to interact with the user, offering a practical way to design command-line tools.

This project can serve as a foundation for further enhancements such as file handling for data persistence, error handling mechanisms, data encryption for security, and even graphical user interfaces (GUI) for improved usability. Overall, the banking system project not only reinforces the understanding of core C programming constructs but also encourages the development of structured and logical thinking, which is essential for any software development task. It is an ideal beginner-level project for students and enthusiasts looking to gain hands-on experience with real-world problem-solving using procedural programming techniques.

REFERENCES

1. **E. Balagurusamy**, *Programming in ANSI C*, McGraw-Hill Education – A comprehensive guide for C programming concepts including structures and dynamic memory allocation.

2. **Brian W. Kernighan and Dennis M. Ritchie**, *The C Programming Language* – The classic and authoritative book on C, covering syntax, data structures, and pointers.

3. **GeeksforGeeks**, https://www.geeksforgeeks.org – Online tutorials and examples on C programming, especially on linked lists, structures, and file handling.

4. **TutorialsPoint**, https://www.tutorialspoint.com/cprogramming/index.htm – A beginner-friendly resource to learn various C concepts and their applications.

5. **Stack Overflow**, https://stackoverflow.com – A helpful platform for resolving coding doubts, debugging, and learning from real-world programming queries.

6. **Cprogramming.com**, https://www.cprogramming.com – Resourceful site for learning and practicing C programming.