Data Structures Innovative Assignment

TOPIC: Bank Management System

Course Code : 2CS302

Made By:

Aayush Kansara (19bce002)

Adhiraj Deshmukh (19bce003)

Agrawal Komal (19bce007)

Project Description:

In this program we tried to build a Bank interface using various functions and data structures.In this program we created one structure to create a data type to store information regarding a bank account like personal details and balance.
We used another structure to create node for a tree storing all the details of a bank account and having a left and right pointer.Using the node structure we created a 2 Binary Search trees for efficient creation , deletion and searching of data entries.
We used the 1st BST to store the bank account details with Account Name as comparison parameter , and used the 2nd BST to store the bank account details with Current Account balance as comparison parameter. By this we can search or edit these account details while only requiring Name , Account balance or account no. details as parameter. And then we created several functions to help create the BSTs , and help create or edit the account details. And functions to display the account details with id Number or Alphabetical order according to name or in descending order of current account balance as parameters using BST.

The Functions Provided to users are:

1. Deposit Money
2. Withdraw Money
3. Check Account Balance
4. Apply For Loan
5. Transfer Money From One User To Another

Project Code:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
int max(int a,int b)
{
   return (a<b)?b:a;
}
/* Bank Structure for storing Information of Bank Users*/
struct Bank {
   char name[20];
   int id;
   double curr,save;
   double returns,debt;
};

struct Bank arr[10];/* Array Of Structure(Bank)*/
int n=0;
/* For Keeping Record of Number of Bank Accounts */
/* Node Structure for Binary Search Tree*/
struct Node {
   char name[20];
   int id;
   double curr,save;
   double returns,debt;
   struct Node* left;
   struct Node* right;
};

struct Node* root1=NULL;
```

```
struct Node* root2=NULL;
/* Creating New Node */
/* In Each Node data of 1 user will be stored in BST*/
struct Node* NEW(struct Bank ob)
{
        struct Node* temp = (struct Node*) malloc(sizeof(struct Node));
        strcpy(temp->name,ob.name);
        temp->id=ob.id;
        temp->curr=ob.curr;temp->save=ob.save;
        temp->returns=ob.returns;temp->debt=ob.debt;
        temp->left=temp->right=NULL;
}
/* Displaying User Information According to Id Number*/
void display1(struct Bank ob)
{
    printf("Name : %s\n",ob.name);
    printf("Account No. : %d\n",ob.id);
    printf("Current Account = %0.3lf , Saving Account =
%0.3lf\n",ob.curr,ob.save);
    printf("Profit Returns = %0.3lf , Debt =
%0.3lf\n\n",ob.returns,ob.debt);
}
/* Displaying User Information According to Name*/
void display2(struct Node* root)
{
    if(!root)
        return;

    printf("Name : %s\n",root->name);
    printf("Account No. : %d\n",root->id);
    printf("Current Account = %0.3lf , Saving Account =
%0.3lf\n",root->curr,root->save);
```

```
    printf("Profit Returns = %0.3lf , Debt = %0.3lf\n\n",root-
>returns,root->debt);
}
/* Comparing the height of left subtree and rightsubtree*/
int depth(struct Node* root)
{
    if(!root)
        return 0;
    return max(depth(root->left),depth(root->right)) + 1;
}
/* Inserting the node in the BST-1*/
struct Node* insert1(struct Node* root,struct Bank ob)
{
    if(!root)
    {
        root = NEW(ob);
        return root;
    }

    if(strcmp(ob.name,root->name) < 0)
        root->left=insert1(root->left,ob);
    else
        root->right=insert1(root->right,ob);

    return root;
}
/* Inserting the node in the BST-2*/
struct Node* insert2(struct Node* root,struct Bank ob)
{
    if(!root)
    {
        root = NEW(ob);
        return root;
```

```
    }

    if(ob.curr < root->curr)
        root->left=insert2(root->left,ob);
    else
        root->right=insert2(root->right,ob);

    return root;
}
/* Adding User Info in BST where each node represents 1 user*/
void add()
{
    if(n == 10)
    {
        printf("Accounts limit reached! You cant create more
Accounts!\n\n");
        return;
    }
 /* Initializing Every variable with zero */
    struct Bank temp;
    strcpy(temp.name,"");
    temp.id=n+1;
    temp.curr=temp.save=temp.returns=temp.debt=0;
   /* Taking Input From User*/
    printf("Name : ");
    fflush(stdin);
    gets(temp.name);

    printf("\nCurrent Account Initial Balance : \n");

    do {
        printf("Enter Value ( >= 0) : ");
        scanf("%lf",&temp.curr);
```

```
    } while(temp.curr < 0);

    printf("\nSaving Account Initial Balance : \n");
    do {
        printf("Enter Value ( >= 5000) : ");
        scanf("%lf",&temp.save);
    } while(temp.save < 5000);
/* Do-While Loop For Checking If Saving Account has minimum
Balance of 5000 else the bank Account Cannot Be Created*/
    printf("\n");

    temp.returns = 0.1*temp.save;

    arr[n]=temp;

    root1=insert1(root1,arr[n]);
    root2=insert2(root2,arr[n]);

    n++;
}
/* Searching for Node In BST-1*/
struct Node* search1(struct Node* root,char name[])
{
    if(!root)
        return NULL;

    int k=strcmp(name,root->name);
    if(k < 0)
        return search1(root->left,name);
    else if(k > 0)
        return search1(root->right,name);

    return root;
```

```
}
/* Searching for node in BST-2*/
struct Node* search2(struct Node* root,int curr)
{
    if(!root)
        return NULL;

    if(curr < root->curr)
        return search2(root->left,curr);
    else if(curr > root->curr)
        return search2(root->right,curr);

    return root;
}
/* Deleting the node in BST-1*/
struct Node* delete_node1(struct Node* root,char name[])
{
    if(!root)
        return NULL;

    int k=strcmp(name,root->name);

    if(k < 0)
    {
        root->left=delete_node1(root->left,name);
        return root;
    }
    else if(k > 0)
    {
        root->right=delete_node1(root->right,name);
        return root;
    }
```

```
    if(!root->left)
    {
        struct Node* temp=root->right;
        root=NULL;
        free(root);
        return temp;
    }
    else if(!root->right)
    {
        struct Node* temp=root->left;
        root=NULL;
        free(root);
        return temp;
    }
//Checking the height of both the subtree For efficient deletion
//If height of left subtree is more then node is deleted from left-
subtree else right-subtree
    if(depth(root->left) > depth(root->right))
    {
        struct Node* temp=root->left;

        while(temp->right)
            temp=temp->right;

        strcpy(root->name,temp->name);
        root->id=temp->id;
        root->curr=temp->curr;root->save=temp->save;
        root->returns=temp->returns;root->debt=temp->debt;

        root->left=delete_node1(root->left,temp->name);
    }
    else
    {
```

```
        struct Node* temp=root->right;

        while(temp->left)
            temp=temp->left;

        strcpy(root->name,temp->name);
        root->id=temp->id;
        root->curr=temp->curr;root->save=temp->save;
        root->returns=temp->returns;root->debt=temp->debt;

        root->right=delete_node1(root->right,temp->name);
    }

    return root;
}
/*  Deletion of node in bst */
struct Node* delete_node2(struct Node* root,int curr)
{
    if(!root)
        return NULL;


    if(curr < root->curr)
    {
        root->left=delete_node2(root->left,curr);
        return root;
    }
    else if(curr > root->curr)
    {
        root->right=delete_node2(root->right,curr);
        return root;
    }
```

```
    if(!root->left)
    {
        struct Node* temp=root->right;
        root=NULL;
        free(root);
        return temp;
    }
    else if(!root->right)
    {
        struct Node* temp=root->left;
        root=NULL;
        free(root);
        return temp;
    }
//Checking the height of both the subtree For efficient deletion
    if(depth(root->left) > depth(root->right))//If height of left subtree is
mor then node is deleed from leftsubtree else right subtree
    {
        struct Node* temp=root->left;

        while(temp->right)
            temp=temp->right;

        strcpy(root->name,temp->name);
        root->id=temp->id;
        root->curr=temp->curr;root->save=temp->save;
        root->returns=temp->returns;root->debt=temp->debt;

        root->left=delete_node2(root->left,temp->curr);
    }
    else
    {
        struct Node* temp=root->right;
```

```
        while(temp->left)
            temp=temp->left;

        strcpy(root->name,temp->name);
        root->id=temp->id;
        root->curr=temp->curr;root->save=temp->save;
        root->returns=temp->returns;root->debt=temp->debt;

        root->right=delete_node2(root->right,temp->curr);
    }

    return root;
}
/* Deleting Node */
void delete_node(int id)
{
    if(id<1 || id>n)
    {
        printf("Invalid Id!\n\n");
        return;
    }

    id--;
    n--;

    root1=delete_node1(root1,arr[id].name);
    root2=delete_node2(root2,arr[id].curr);

    for(int i=id;i<n;i++)
        arr[i]=arr[i+1];
}
//Printing Bst-1 In Inorder Traversal
```

```
void inorder1(struct Node* root)
{
    if(!root) return;
    inorder1(root->left);
    display2(root);
    inorder1(root->right);
}
//Printing Bst-2 In Inorder Traversal
void inorder2(struct Node* root)
{
    if(!root) return;
    inorder2(root->right);
    display2(root);
    inorder2(root->left);
}

void withdraw(int id, int val)
{
    if(id<1 || id>n)
    {
        printf("Invalid Id!\n\n");
        return;
    }

    id--;

    int choice = 0;
    printf("Current Account = 1 , Savings Account = 2\n");
    printf("Option : ");
    scanf("%d",&choice);
    printf("\n");
  //User will Select The Account From Which He wants to withdraw
money Savings or current
```

```
    if(choice == 1)
    {
        if(val > arr[id].curr)
        {
            printf("You cannot withdraw more than your Current balance
!\n\n");
            return;
        }
    //deleting the current node
        root1=delete_node1(root1,arr[id].name);
        root2=delete_node2(root2,arr[id].curr);

        arr[id].curr-=val;
     //Updating the value
        root1=insert1(root1,arr[id]);
        root2=insert2(root2,arr[id]);
//Again inserting these 2 nodes
    }
    else if(choice == 2)
    {
        if(val > arr[id].save-5000)
        {
            printf("You have to keep a minimum balance of 5000 in your
Savings account , you cannot withdraw more than that !\n\n");
            return;
        }

        root1=delete_node1(root1,arr[id].name);
        root2=delete_node2(root2,arr[id].curr);

        arr[id].save-=val;
        arr[id].returns=0.1*arr[id].save;
```

```
        root1=insert1(root1,arr[id]);
        root2=insert2(root2,arr[id]);
    }
    else
    {
        printf("Invalid Option !\n\n");
        return;
    }

    printf("Transaction Successful !\n\n");
}
/*Function To Deposit Money */
void deposit(int id,int val)
{
    if(id<1 || id>n)
    {
        printf("Invalid Id!\n");
        return;
    }

    id--;

    int choice = 0;
//Two  Options Are Provided to user for Money Deposit Savings And
Current
    printf("Current Account = 1 , Savings Account = 2\n");
    printf("Option : ");
    scanf("%d",&choice);
    printf("\n");

    if(choice == 1)
    {
        root1=delete_node1(root1,arr[id].name);
```

```
        root2=delete_node2(root2,arr[id].curr);

        arr[id].curr+=val;

        root1=insert1(root1,arr[id]);
        root2=insert2(root2,arr[id]);
    }
    else if(choice == 2)
    {
        root1=delete_node1(root1,arr[id].name);
        root2=delete_node2(root2,arr[id].curr);

        arr[id].save+=val;
        arr[id].returns=0.1*arr[id].save;

        root1=insert1(root1,arr[id]);
        root2=insert2(root2,arr[id]);
    }
    else
    {
        printf("Invalid Option !\n\n");
        return;
    }

    printf("Deposit Successful !\n\n");
}
/* Function To Check Balance*/
void balance(int id)
{
    if(id<1 || id>n)
    {
        printf("Invalid Id!\n\n");/* */
        return;
```

```
    }

    id--;
    printf("Name : %s\n",arr[id].name);
    printf("Current Balance : %0.3lf\n",arr[id].curr);
    printf("Savings Balance : %0.3lf\n\n",arr[id].save);
}
/* Function For Checking If User is eligible for loan or not (Criteria--
>Loan can be taken for amount>=1000)*/
void loan(int id,int val)
{
    if(id<1 || id>n)/* Out Of Bound*/
    {
        printf("Invalid Id!\n\n");
        return;
    }
    else if(val < 1000)
    {
        printf("Loan only valid for value >=1000 !\n\n");
        return;
    }

    id--;
     /*Deleting the Node So that Bst is Maintained */
    root1=delete_node1(root1,arr[id].name);
    root2=delete_node2(root2,arr[id].curr);

    arr[id].curr+=val;/* Inserting Money In Current Account*/
    arr[id].debt+=val*1.15;/* Loan is available at 15%interest P.A.*/
    /* Inserting The Node Again*/
    root1=insert1(root1,arr[id]);
    root2=insert2(root2,arr[id]);
```

```
    printf("Amount transferred to your current account!\n");
    printf("Total Current Debt : %lf\n\n",arr[id].debt);
}
/* Money Transfer From One Account To Another*/
void transfer(int a,int b,int val) {
    if(a>n || b>n || a<1 || b<1)
    {
        printf("Invalid Account Nos. !\n\n");
        return;
    }

    a--;b--;

    if(val > arr[a].curr)
    {
        printf("Value to be transferred is greater than Current Account
Balance of %d !\n\n",a+1);
        return;
    }
/*Deleting Both the nodes */
    root1=delete_node1(root1,arr[a].name);
    root2=delete_node2(root2,arr[a].curr);

    root1=delete_node1(root1,arr[b].name);
    root2=delete_node2(root2,arr[b].curr);
/* Updating the values of current node*/
    arr[a].curr-=val;
    arr[b].curr+=val;
/* Inserting The Updated Nodes*/
    root1=insert1(root1,arr[a]);
    root2=insert2(root2,arr[a]);

    root1=insert1(root1,arr[b]);
```

```
    root2=insert2(root2,arr[b]);

    printf("Money Transfer Successful from Account %d to Account
%d !\n\n",a+1,b+1);
}
/* Main Function */
int main()
{
    printf("Welcome to Bank/ATM Interface !\n\n");
    int choice=0;
    while(choice!=6)
    {
        printf("************************\n");
        printf("1. Create New Account\n");
        printf("2. Choose Account\n");
        printf("3. Display All Accounts Details in ID Order\n");
        printf("4. Display All Accounts Details in Naming Order\n");
        printf("5. Display All Accounts Details in Descending Order of
Current Balance\n");
        printf("6. Exit\n");
        printf("************************\n\n");
        //outer do while loop for creating account,selecting account
and displaying account
        printf("Option 1 : ");
        scanf("%d",&choice);

        printf("\n");

        if(choice == 1)
            add();
        else if(choice == 2)
        {
            if(!n)
```

```
{
    printf("No Accounts Available !\n\n");
    continue;
}

printf("Choose account : \n");
int x = 0;
do {
    printf("Select from (1 - %d) : ",n);
    scanf("%d",&x);
} while(x<1 && x>n);

x--;

printf("\n************************\n");
printf("(1) Withdraw\n");
printf("(2) Check Balance\n");
printf("(3) Deposit\n");
printf("(4) Loan\n");
printf("(5) Money Transfer\n");
printf("************************\n\n");
//inner do while loop for User Functionalities
printf("Option 2 : ");
int choice2;
scanf("%d",&choice2);

int val=0;
if(choice2 == 1)
{
    printf("\nEnter the amount to be Withdrawn : \n");
    val=0;

    do {
```

```
        printf("Enter Value ( >= 0) : ");
        scanf("%d",&val);
    } while(val<0);

    printf("\n");

    withdraw(x+1,val);
}
else if(choice2 == 2)
{
    printf("Account %d Balance : \n\n",x+1);
    balance(x+1);
}
else if(choice2 == 3)
{
    printf("\nEnter the amount to be Deposited : \n");
    val=0;

    do {
        printf("Enter Value ( >= 0) : ");
        scanf("%d",&val);
    } while(val<0);

    printf("\n");

    deposit(x+1,val);
}
else if(choice2 == 4)
{
    printf("\nEnter the amount to be Lent : \n");
    val=0;

    do {
```

```
            printf("Enter Value >=0 : ");
            scanf("%d",&val);
        } while(val<0);

        printf("\n");

        loan(x+1,val);
    }
    else if(choice2 == 5)
    {
        printf("\nEnter Account no. : \n");
        int y=0;

        do {
            printf("Select from (1 - %d) : ",n);
            scanf("%d",&y);
        } while(y<1 && y>n);

        printf("\nEnter the amount to be Transferred : \n");
        val=0;

        do {
            printf("Enter Value >=0 : ");
            scanf("%d",&val);
        } while(val<0);

        printf("\n");

        transfer(x+1,y,val);
    }
    else
        printf("Invalid Option 2 !\n\n");
}
```

```
        else if(choice == 3)
        {
            for(int i=0;i<n;i++)
                display1(arr[i]);
        }
        else if(choice == 4)
            inorder1(root1);
        else if(choice == 5)
            inorder2(root2);
        else if(choice == 6)
        {
            printf("Exiting .... \n\n");
            break;
        }
        else
            printf("Invalid Option 1 !\n\n");
    }

    return 0;
}
```

OUTPUT:

```
Welcome to Bank/ATM Interface !

**************************
1. Create New Account
2. Choose Account
3. Display All Accounts Details in ID Order
4. Display All Accounts Details in Naming Order
5. Display All Accounts Details in Descending Order of Current Balance
6. Exit
**************************

Option 1 : 1

Name : Aayush

Current Account Initial Balance :
Enter Value ( >= 0) : 45

Saving Account Initial Balance :
Enter Value ( >= 5000) : 6548

**************************
1. Create New Account
2. Choose Account
3. Display All Accounts Details in ID Order
4. Display All Accounts Details in Naming Order
5. Display All Accounts Details in Descending Order of Current Balance
6. Exit
**************************

Option 1 : 1

Name : Bharat

Current Account Initial Balance :
Enter Value ( >= 0) : 55

Saving Account Initial Balance :
Enter Value ( >= 5000) : 7845

**************************
1. Create New Account
2. Choose Account
3. Display All Accounts Details in ID Order
4. Display All Accounts Details in Naming Order
5. Display All Accounts Details in Descending Order of Current Balance
6. Exit
**************************

Option 1 : 1

Name : Komal
```

C:\Users\infinite\Desktop\Dsa_Assignment.exe

```
Option 1 : 1

Name : Komal

Current Account Initial Balance :
Enter Value ( >= 0) : 687

Saving Account Initial Balance :
Enter Value ( >= 5000) : 45798

**************************
1. Create New Account
2. Choose Account
3. Display All Accounts Details in ID Order
4. Display All Accounts Details in Naming Order
5. Display All Accounts Details in Descending Order of Current Balance
6. Exit
**************************

Option 1 : 1

Name : Vinit

Current Account Initial Balance :
Enter Value ( >= 0) : 456

Saving Account Initial Balance :
Enter Value ( >= 5000) : 7878

**************************
1. Create New Account
2. Choose Account
3. Display All Accounts Details in ID Order
4. Display All Accounts Details in Naming Order
5. Display All Accounts Details in Descending Order of Current Balance
6. Exit
**************************

Option 1 : 1

Name : Nitin

Current Account Initial Balance :
Enter Value ( >= 0) : 65

Saving Account Initial Balance :
Enter Value ( >= 5000) : 8874

**************************
```

C:\Users\infinite\Desktop\Dsa_Assignment.exe

```
Option 1 : 2

Choose account :
Select from (1 - 5) : 2

**************************
(1) Withdraw
(2) Check Balance
(3) Deposit
(4) Loan
(5) Money Transfer
**************************

Option 2 : 1

Enter the amount to be Withdrawn :
Enter Value ( >= 0) : 45

Current Account = 1 , Savings Account = 2
Option : 2

Transaction Successful !

**************************
1. Create New Account
2. Choose Account
3. Display All Accounts Details in ID Order
4. Display All Accounts Details in Naming Order
5. Display All Accounts Details in Descending Order of Current Balance
6. Exit
**************************

Option 1 : 2

Choose account :
Select from (1 - 5) : 5

**************************
(1) Withdraw
(2) Check Balance
(3) Deposit
(4) Loan
(5) Money Transfer
**************************

Option 2 : 3

Enter the amount to be Deposited :
Enter Value ( >= 0) : 500

Current Account = 1 , Savings Account = 2
Option : 2

Deposit Successful !

**************************
```

C:\Users\infinite\Desktop\Dsa_Assignment.exe

```
4. Display All Accounts Details in Naming Order
5. Display All Accounts Details in Descending Order of Current Balance
6. Exit
**************************

Option 1 : 2

Choose account :
Select from (1 - 5) : 5

**************************
(1) Withdraw
(2) Check Balance
(3) Deposit
(4) Loan
(5) Money Transfer
**************************

Option 2 : 2
Account 5 Balance :

Name : Nitin
Current Balance : 65.000
Savings Balance : 9374.000

**************************
1. Create New Account
2. Choose Account
3. Display All Accounts Details in ID Order
4. Display All Accounts Details in Naming Order
5. Display All Accounts Details in Descending Order of Current Balance
6. Exit
**************************

Option 1 : 2

Choose account :
Select from (1 - 5) : 3

**************************
(1) Withdraw
(2) Check Balance
(3) Deposit
(4) Loan
(5) Money Transfer
**************************

Option 2 : 5

Enter Account no. :
Select from (1 - 5) : 2

Enter the amount to be Transferred :
Enter Value >=0 : 45

Money Transfer Successful from Account 3 to Account 2 !
```

C:\Users\infinite\Desktop\Dsa_Assignment.exe

```
Option 1 : 3

Name : Aayush
Account No. : 1
Current Account = 45.000 , Saving Account = 6548.000
Profit Returns = 654.800 , Debt = 0.000

Name : Bharat
Account No. : 2
Current Account = 100.000 , Saving Account = 7800.000
Profit Returns = 780.000 , Debt = 0.000

Name : Komal
Account No. : 3
Current Account = 642.000 , Saving Account = 45798.000
Profit Returns = 4579.800 , Debt = 0.000

Name : Vinit
Account No. : 4
Current Account = 456.000 , Saving Account = 7878.000
Profit Returns = 787.800 , Debt = 0.000

Name : Nitin
Account No. : 5
Current Account = 65.000 , Saving Account = 9374.000
Profit Returns = 937.400 , Debt = 0.000

**************************
1. Create New Account
2. Choose Account
3. Display All Accounts Details in ID Order
4. Display All Accounts Details in Naming Order
5. Display All Accounts Details in Descending Order of Current Balance
6. Exit
**************************

Option 1 : 4

Name : Aayush
Account No. : 1
Current Account = 45.000 , Saving Account = 6548.000
Profit Returns = 654.800 , Debt = 0.000

Name : Bharat
Account No. : 2
Current Account = 100.000 , Saving Account = 7800.000
Profit Returns = 780.000 , Debt = 0.000

Name : Komal
Account No. : 3
Current Account = 642.000 , Saving Account = 45798.000
Profit Returns = 4579.800 , Debt = 0.000

Name : Nitin
Account No. : 5
Current Account = 65.000 , Saving Account = 9374.000
Profit Returns = 937.400 , Debt = 0.000

Name : Vinit
```

C:\Users\infinite\Desktop\Dsa_Assignment.exe

```
Profit Returns = 4579.800 , Debt = 0.000

Name : Nitin
Account No. : 5
Current Account = 65.000 , Saving Account = 9374.000
Profit Returns = 937.400 , Debt = 0.000

Name : Vinit
Account No. : 4
Current Account = 456.000 , Saving Account = 7878.000
Profit Returns = 787.800 , Debt = 0.000

**************************
1. Create New Account
2. Choose Account
3. Display All Accounts Details in ID Order
4. Display All Accounts Details in Naming Order
5. Display All Accounts Details in Descending Order of Current Balance
6. Exit
**************************

Option 1 : 5

Name : Komal
Account No. : 3
Current Account = 642.000 , Saving Account = 45798.000
Profit Returns = 4579.800 , Debt = 0.000

Name : Vinit
Account No. : 4
Current Account = 456.000 , Saving Account = 7878.000
Profit Returns = 787.800 , Debt = 0.000

Name : Bharat
Account No. : 2
Current Account = 100.000 , Saving Account = 7800.000
Profit Returns = 780.000 , Debt = 0.000

Name : Nitin
Account No. : 5
Current Account = 65.000 , Saving Account = 9374.000
Profit Returns = 937.400 , Debt = 0.000

Name : Aayush
Account No. : 1
Current Account = 45.000 , Saving Account = 6548.000
Profit Returns = 654.800 , Debt = 0.000

**************************
1. Create New Account
2. Choose Account
3. Display All Accounts Details in ID Order
4. Display All Accounts Details in Naming Order
5. Display All Accounts Details in Descending Order of Current Balance
6. Exit
**************************

Option 1 : 6
```