```cpp
#include <iostream>
#include <string>
using namespace std;

const int MAX_AVAILABLE_ACCOUNTS = 3;

struct BankAccount
{
    unsigned int AccountNumber;
    bool AccountType;
    string AccountHolderName;
    string CNIC; // It is better to keep the CNIC as a string rather than a char array
    double CurrentBalance;
    unsigned int PIN;
};

// Global variables (can be accessed by any function)
// Here we are telling the program that BankAccount is an array of size MAX_AVAILABLE_ACCOUNTS
BankAccount CustomerAccount[MAX_AVAILABLE_ACCOUNTS];
int accountCount = 0;

// Some of the parameters in these functions are const bcz we do not want them to change
// Some of the parameters in these functions are not const bcz they change the account info

// Function to display the title
void Title();
// Function to display an options menu and let customer choose an option
unsigned int OptionsMenu();
// Function to make a new account
void OpenAccount(BankAccount CustomerAccount[MAX_AVAILABLE_ACCOUNTS], int& accountCount);
// Function to deposit ammount into the customer's account (if he enters the AccNo and PIN correct)
void Deposit(BankAccount CustomerAccount[MAX_AVAILABLE_ACCOUNTS], int accountCount);
// Function to withdraw ammount from the customer's account (if he enters the AccNo and PIN correct)
void Withdraw(BankAccount CustomerAccount[MAX_AVAILABLE_ACCOUNTS], int accountCount);
// Function to display account info of the customer's account (if he enters the AccNo and PIN correct)
void DisplayAccountDetails(const BankAccount CustomerAccount[MAX_AVAILABLE_ACCOUNTS], int accountCount);
// Function to display the account info of all of the customers
void DisplayAllCustomersAccountsDetails(const BankAccount CustomerAccount
```

```
    [MAX_AVAILABLE_ACCOUNTS], int accountCount);
// Function to search for an account by entering account number
int SearchAccount(const BankAccount CustomerAccount
    [MAX_AVAILABLE_ACCOUNTS], int accountCount, int AccountNumber);
// Function to make sure the CNIC format is correct
bool isValidCNIC(const string& CNIC);
// Ending line
void EndLine();

int main()
{
    // Variables
    unsigned int choice = 6;
    int accountIndex = -1; // Initialized accountIndex before the switch
      statement
                           // Reason: If the variable initialization
                    appears after the switch statement or isn't properly
                    handled in the default case,
                           // it can lead to situations where the variable
                     might not be initialized when the default case
                    executes.
    // Display the title
    Title();
    // Loop to keep asking for an input if the user enters invalid input

    while (true)
    {
        // Display the options menu and ask user for a choice
        choice = OptionsMenu();

        // Call the necessary function depending on what the user has
          choosen
        switch (choice)
        {
        case 1:
            OpenAccount(CustomerAccount, accountCount);
            break;
        case 2:
            Deposit(CustomerAccount, accountCount);
            break;
        case 3:
            Withdraw(CustomerAccount, accountCount);
            break;
        case 4:
            int accountNumber;
            cout << "Enter the account number to search: ";
            cin >> accountNumber;
            cout << endl;
            // Calling SearchAccount and checking if the account is found
```

```
80              accountIndex = SearchAccount(CustomerAccount, accountCount,
                  accountNumber);
81              cout << "\n\n";
82              if (accountIndex != -1) //Acc exists
83              {
84                  DisplayAccountDetails(CustomerAccount, accountCount);
85              }
86              else
87              {
88                  cout << "!!! Account not found !!!" << endl;
89              }
90              break;
91          case 5:
92              DisplayAccountDetails(CustomerAccount, accountCount);
93              break;
94          case 6:
95              DisplayAllCustomersAccountsDetails(CustomerAccount,
                  accountCount);
96              break;
97          case 7:
98              cout << "Thank you for using our services. If you need help,
                  please contact support at support@bank.com" << endl;
99              return 0;
100         default:
101             cout << "Invalid choice. Please try again!" << endl;
102             continue;
103         }
104     }
105     return 0;
106 }
107
108 void Title()
109 {
110     cout << "\"Welcome to the Bank Al-PIEAS Limited!\"\n";
111 }
112
113 unsigned int OptionsMenu()
114 {
115     unsigned int choice = 6;
116
117     cout << "\n========== Bank Menu ==========\n";
118     cout << "-------------------------------\n";
119     cout << "| 1. Open New Account          |\n";
120     cout << "| 2. Deposit                   |\n";
121     cout << "| 3. Withdraw                  |\n";
122     cout << "| 4. Search Account            |\n";
123     cout << "| 5. Display Account Details   |\n";
124     cout << "| 6. Display All Accounts Info |\n";
125     cout << "| 7. Exit                      |\n";
```

```
126     cout << "------------------------------\n\n";
127     cout << "Please enter your choice (1-7): ";
128     cin >> choice;
129     cout << endl;
130     return choice;
131 }
132
133 void OpenAccount(BankAccount CustomerAccount[MAX_AVAILABLE_ACCOUNTS], int& accountCount)
134 {
135     if (accountCount >= MAX_AVAILABLE_ACCOUNTS)
136     {
137         cout << "Account limit reached. Cannot open new account." << endl;
138         return; // return statement exits the loop
139     }
140
141     // TAKING IN THE ACCOUNT TYPE
142     cout << "Select the account type (Savings/Current):\n";
143     while (true)
144     {
145         cout << "Press 0 for Current OR 1 for Savings: ";
146         cin >> CustomerAccount[accountCount].AccountType;
147         if (CustomerAccount[accountCount].AccountType != 0 &&
              CustomerAccount[accountCount].AccountType != 1)
148         {
149             cout << "Invalid Input! You can enter either 0 or 1.\n";
150         }
151         else
152         {
153             break;
154         }
155     }
156     cout << endl;
157
158     //TAKING IN THE ACCOUNT HOLDER'S NAME
159     cout << "Enter Account Holder's Name: ";
160     cin.ignore();  // To ignore the newline character from previous input
161     getline(cin, CustomerAccount[accountCount].AccountHolderName);
162     cout << endl;
163
164     // TAKING IN THE ACCOUND HOLDER'S CNIC
165     while (true)
166     {
167         cout << "Enter CNIC (format: *****-*******-*): ";
168         getline(cin, CustomerAccount[accountCount].CNIC);
169
170         // Validate CNIC format
171         if (!isValidCNIC(CustomerAccount[accountCount].CNIC))
172         {
```

```
173             cout << "\nInvalid format!\n";
174         }
175         else // Valid input
176         {
177             break;
178         }
179         cout << endl;
180     }
181     cout << endl;
182     // TAKING IN THE ACCOUND HOLDER'S PIN
183     while (true)
184     {
185         cout << "Please set 4-digit PIN. (This PIN will be used to access
              your acccount): ";
186         cin >> CustomerAccount[accountCount].PIN;
187         cout << endl;
188         // validate PIN length
189         if (CustomerAccount[accountCount].PIN < 1000 || CustomerAccount
              [accountCount].PIN > 9999)
190         {
191             cout << "Error! The PIN you have entered is not of 4-digits:
                  \n";
192             cin.clear();
193             cin.ignore(10000, '\n');
194             cout << endl;
195         }
196         else
197         {
198             cout << "Your PIN has been created.\Do not share your PIN with
                   anyone!\n";
199             break;
200         }
201     }
202     cout << endl;
203     // TAKING IN THE ACCOUND HOLDER'S INITIAL BALANCE
204     while (true)
205     {
206         cout << "Enter Initial Balance (minimum Rs. 5000/-): ";
207         cin >> CustomerAccount[accountCount].CurrentBalance;
208         cout << endl;
209         // validate initial amount deposited
210         if (CustomerAccount[accountCount].CurrentBalance < 5000)
211         {
212             cout << "Invalid amount!\n";
213             cin.clear();
214             cin.ignore(5000, '\n');
215         }
216         else
217         {
```

```
218             cout << "Amount has been added to you account.\n\nCurrent
                  amount in your account is: Rs. ";
219             cout << CustomerAccount[accountCount].CurrentBalance;
220             cout << endl;
221             break;
222         }
223     }
224     cout << endl;
225     // ALLOTT ACCOUNT NUMBER
226     CustomerAccount[accountCount].AccountNumber = 5995801937 +
          accountCount;
227     accountCount++;  // Increment account count after assignment
228     cout << "Congratulations! Your bank account has been created.\n\nYour
          Account Number is " << CustomerAccount[accountCount -
          1].AccountNumber << ". ";
229     cout << "Remember it for future reference.\n";
230     EndLine;
231 }
232
233 void Deposit(BankAccount CustomerAccount[MAX_AVAILABLE_ACCOUNTS], int
      accountCount)
234 {
235     int accountNumber;
236     int index = -1;
237     int retryCount = 0; // Counter for retry attempts for account number
238     int pinRetryCount = 0; // Counter for retry attempts for PIN
239
240     while (true)
241     {
242         cout << "Enter your account number (10 digits): ";
243         cin >> accountNumber;
244         cout << endl;
245         // Validate the length of the account number
246         if (accountNumber < 1000000000 || accountNumber > 9999999999)
247         {
248             cout << "Invalid no of digits!\n\n";
249             cin.clear();
250             cin.ignore(10000, '\n');
251         }
252         else
253         {
254             // Search for the account using the account number
255             index = SearchAccount(CustomerAccount, accountCount,
                      accountNumber);
256
257             // If account is not found
258             if (index == -1)
259             {
260                 cout << "!!! Account not found !!!\n\n";
```

```
261
262                 // Give user a choice whether they want to try again or go to main menu
263                 char choice;
264                 cout << "Would you like to try again? (Y/N): ";
265                 cin >> choice;
266                 cout << endl;
267
268                 if (choice == 'Y' || choice == 'y')
269                 {
270                     retryCount++; // Increment retry count
271                     if (retryCount >= 3) // Limit retries to 3
272                     {
273                         cout << "Too many failed attempts. Returning to main menu.\n";
274                         return; // Exit after 3 failed attempts
275                     }
276                     else
277                     {
278                         cout << "You have " << (3 - retryCount) << " attempt(s) remaining.\n\n";
279                         continue; // Restart the loop to enter account number again
280                     }
281                 }
282                 else if (choice == 'N' || choice == 'n')
283                 {
284                     return; // Exit the function and go back to the main menu
285                 }
286                 else
287                 {
288                     cout << "Invalid choice! Returning to main menu.\n";
289                     return; // Exit the function for an invalid choice as well
290                 }
291             }
292             else // Account is found
293             {
294                 break; // Exit the loop if account is found
295             }
296         }
297     }
298     cout << endl;
299     // PIN retry logic (limit to 3 attempts)
300     unsigned int pin;
301     while (pinRetryCount < 3) // Allow up to 3 PIN attempts
302     {
303         cout << "Enter you 4-digit PIN: ";
```

```
304         cin >> pin;
305         cout << endl;
306
307         if (pin != CustomerAccount[index].PIN)
308         {
309             pinRetryCount++; // Increment PIN retry count
310             cout << "!!! Incorrect PIN !!!\n";
311
312             if (pinRetryCount >= 3) // Limit retries to 3
313             {
314                 cout << "Too many incorrect PIN attempts. Returning to
                      main menu.\n";
315                 return; // Exit after 3 failed PIN attempts
316             }
317             cout << endl;
318             cout << "You have " << (3 - pinRetryCount) << " attempt(s)
                  remaining.\n";
319         }
320         else
321         {
322             break; // Exit loop if PIN is correct
323         }
324         cout << endl;
325     }
326
327     // If the PIN is correct, ask for deposit amount
328     double amount;
329     cout << "Enter deposit amount: ";
330     while (!(cin >> amount) || amount <= 0)
331     {
332         cout << "Invalid amount! Enter a positive number: ";
333         cin.clear();
334         cin.ignore(10000, '\n');
335     }
336
337     // Add the deposit to the account balance
338     CustomerAccount[index].CurrentBalance += amount;
339     cout << "Rs. " << amount << " deposited. New balance: " <<
          CustomerAccount[index].CurrentBalance << endl;
340     EndLine;
341 }
342
343 void Withdraw(BankAccount CustomerAccount[MAX_AVAILABLE_ACCOUNTS], int
      accountCount)
344 {
345     int accountNumber;
346     int index = -1;
347     int retryCount = 0; // Counter for retry attempts for account number
348     int pinRetryCount = 0; // Counter for retry attempts for PIN
```

```
349
350     while (true)
351     {
352         cout << "Enter your account number (10 digits): ";
353         cin >> accountNumber;
354         cout << endl;
355
356         // Validate the length of the account number
357         if (accountNumber < 1000000000 || accountNumber > 9999999999)
358         {
359             cout << "Invalid digits length! Enter a 10-digit account
                  number: ";
360             cin.clear();
361             cin.ignore(10000, '\n');
362         }
363         else
364         {
365             // Search for the account using the account number
366             index = SearchAccount(CustomerAccount, accountCount,
                  accountNumber);
367             cout << endl;
368             // If account is not found
369             if (index == -1)
370             {
371                 cout << "Account not found!\n";
372
373                 // Give user a choice whether they want to try again or go
                     to main menu
374                 char choice;
375                 cout << "Would you like to try again? (Y/N): ";
376                 cin >> choice;
377
378                 if (choice == 'Y' || choice == 'y')
379                 {
380                     retryCount++; // Increment retry count
381                     if (retryCount >= 3) // Limit retries to 3
382                     {
383                         cout << "Too many failed attempts. Returning to
                        main menu.\n";
384                         return; // Exit after 3 failed attempts
385                     }
386                     else
387                     {
388                         cout << "You have " << (3 - retryCount) << "
                        attempt(s) remaining.\n\n";
389                         continue; // Restart the loop to enter account
                        number again
390                     }
391                 }
```

```
392                 else if (choice == 'N' || choice == 'n')
393                 {
394                     return; // Exit the function and go back to the main menu
395                 }
396                 else
397                 {
398                     cout << "Invalid choice! Returning to main menu.\n";
399                     return; // Exit the function for an invalid choice as well
400                 }
401             }
402             else // Account is found
403             {
404                 break; // Exit the loop if account is found
405             }
406         }
407     }
408     cout << endl;
409 
410     unsigned int pin;
411     while (pinRetryCount < 3) // Allow up to 3 PIN attempts
412     {
413         cout << "Enter your 4-digit PIN: ";
414         cin >> pin;
415         cout << endl;
416 
417         if (pin != CustomerAccount[index].PIN)
418         {
419             pinRetryCount++; // Increment PIN retry count
420             cout << "Incorrect PIN!\n";
421 
422             if (pinRetryCount >= 3) // Limit retries to 3
423             {
424                 cout << "Too many incorrect PIN attempts. Returning to main menu.\n";
425                 return; // Exit after 3 failed PIN attempts
426             }
427             cout << endl;
428             cout << "You have " << (3 - pinRetryCount) << " attempt(s) remaining.\n";
429         }
430         else
431         {
432             break; // Exit loop if PIN is correct
433         }
434         cout << endl;
435     }
436 
```

```
437     double amount;
438     while (true)
439     {
440         cout << "Enter withdrawal amount: ";
441         cin >> amount;
442         cout<<endl;
443 
444         if (amount <= 0)
445         {
446             cout << "Invalid amount! Enter a positive number: ";
447             cin.clear();
448             cin.ignore(10000, '\n');
449         }
450         else
451         {
452             // Check if withdrawal amount exceeds balance
453             if (amount > CustomerAccount[index].CurrentBalance)
454             {
455                 cout << "Insufficient balance! You cannot withdraw more than your current balance.\n\n";
456 
457                 // Ask the user if they want to try again or go back to the main menu
458                 char choice;
459                 cout << "Would you like to try again or return to main menu? (T for Try Again / M for Main Menu): ";
460                 cin >> choice;
461                 cout << endl;
462 
463                 if (choice == 'T' || choice == 't')
464                 {
465                     Withdraw(CustomerAccount, accountCount);  // Recursively call the Withdraw function to try again
466                 }
467                 else if (choice == 'M' || choice == 'm')
468                 {
469                     return; // Exit the function and go back to the main menu
470                 }
471                 else
472                 {
473                     cout << "Invalid choice! Returning to main menu.\n";
474                     return; // Exit the function for an invalid choice
475                 }
476             }
477         }
478         break;
479     }
480 
```

```
481     // If withdrawal is valid, subtract from balance
482     CustomerAccount[index].CurrentBalance -= amount;
483     cout << "Rs. " << amount << "/- have been withdrawn from your account.
           Remaining balance: " << CustomerAccount[index].CurrentBalance <<
          "/-" << endl;
484     EndLine;
485 }
486
487 int SearchAccount(const BankAccount CustomerAccount
      [MAX_AVAILABLE_ACCOUNTS], int accountCount, int accountNumber)
488 {
489     for (int i = 0; i < accountCount; i++) // Loop through all accounts
490     {
491         if (CustomerAccount[i].AccountNumber == accountNumber) // Check if
              account number matches
492         {
493             cout << "Your account exists in our bank branch.\n\n";
494             cout << "Account Holder: " << CustomerAccount
                  [i].AccountHolderName << endl;
495             return i; // Return the index of the found account
496         }
497     }
498     return -1; // Account not found, return -1
499 }
500
501 void DisplayAccountDetails(const BankAccount CustomerAccount
      [MAX_AVAILABLE_ACCOUNTS], int accountCount)
502 {
503     int accountNumber;
504     int index = -1;
505     int retryCount = 0; // Counter for retry attempts for account number
506     int pinRetryCount = 0; // Counter for retry attempts for PIN
507
508         while (true)
509         {
510             cout << "Enter your account number (10 digits): ";
511             cin >> accountNumber;
512
513             // Validate the length of the account number
514             if (accountNumber < 1000000000 || accountNumber > 9999999999)
515             {
516                 cout << "Invalid digits length! Enter a 10-digit account
                      number: ";
517                 cin.clear();
518                 cin.ignore(10000, '\n');
519             }
520             else
521             {
522                 // Search for the account using the account number
```

```
523                 index = SearchAccount(CustomerAccount, accountCount,
                      accountNumber);
524
525                 // If account is not found
526                 if (index == -1)
527                 {
528                     cout << "Account not found!\n";
529
530                     // Give user a choice whether they want to try again
                    or go to main menu
531                     char choice;
532                     cout << "Would you like to try again? (Y/N): ";
533                     cin >> choice;
534
535                     if (choice == 'Y' || choice == 'y')
536                     {
537                         retryCount++; // Increment retry count
538                         if (retryCount >= 3) // Limit retries to 3
539                         {
540                             cout << "Too many failed attempts. Returning
                    to main menu.\n";
541                             return; // Exit after 3 failed attempts
542                         }
543                         continue; // Restart the loop to enter account
                    number again
544                     }
545                     else if (choice == 'N' || choice == 'n')
546                     {
547                         return; // Exit the function and go back to the
                    main menu
548                     }
549                     else
550                     {
551                         cout << "Invalid choice! Returning to main menu.
                    \n";
552                         return; // Exit the function for an invalid choice
                     as well
553                     }
554                 }
555                 else // Account is found
556                 {
557                     break; // Exit the loop if account is found
558                 }
559             }
560         }
561
562     unsigned int pin;
563     while (pinRetryCount < 3) // Allow up to 3 PIN attempts
564     {
```

```
565         cout << "Enter your 4-digit PIN: ";
566         cin >> pin;
567
568         // Validate that the PIN matches the account
569         if (pin != CustomerAccount[index].PIN)
570         {
571             pinRetryCount++; // Increment PIN retry count
572             cout << "Incorrect PIN for the given account number!\n";
573
574             if (pinRetryCount >= 3) // Limit retries to 3
575             {
576                 cout << "Too many incorrect PIN attempts. Returning to
                      main menu.\n";
577                 return; // Exit after 3 failed PIN attempts
578             }
579             cout << "You have " << (3 - pinRetryCount) << " attempt(s)
                  remaining.\n";
580         }
581         else
582         {
583             break; // Exit loop if PIN matches the account
584         }
585     }
586
587     // Display account details after successful validation
588     cout << "\nAccount Details:-" << endl;
589     cout << "-----------------------------" << endl;
590     cout << "Account Number: " << CustomerAccount[index].AccountNumber <<
        endl;
591     cout << "Account Type: " << (CustomerAccount[index].AccountType ?
        "Savings" : "Current") << endl;
592     cout << "Account Holder: " << CustomerAccount[index].AccountHolderName
         << endl;
593     cout << "CNIC: " << CustomerAccount[index].CNIC << endl;
594     cout << "Balance: Rs. " << CustomerAccount[index].CurrentBalance <<
        endl;
595     cout << "-----------------------------" << endl;
596     cout << "Thank you for using our service.\n";
597 }
598
599 void DisplayAllCustomersAccountsDetails(const BankAccount CustomerAccounts
      [MAX_AVAILABLE_ACCOUNTS], int accountCount)
600 {
601     const unsigned int EMPLOYEE_PIN = 1234; // Employee PIN
602     unsigned int enteredPin;
603     int retryCount = 0; // Retry counter for PIN attempts
604
605     // Ask the employee for the PIN
606     while (retryCount < 3) // Allow up to 3 attempts
```

```
607     {
608         cout << "Enter the Employee PIN: {The employee PIN is 1234 ;-) }";
609         cin >> enteredPin;
610 
611         if (enteredPin == EMPLOYEE_PIN)
612         {
613             // If PIN is correct, display all customer account details
614             cout << "\nEmployee PIN verified successfully.\n";
615             for (int i = 0; i < accountCount; i++)
616             {
617                 cout << "\nAccount Number: " << CustomerAccounts
                      [i].AccountNumber << endl;
618                 cout << "Account Type: " << (CustomerAccounts
                      [i].AccountType ? "Savings" : "Current") << endl;
619                 cout << "Account Holder: " << CustomerAccounts
                      [i].AccountHolderName << endl;
620                 cout << "CNIC: " << CustomerAccounts[i].CNIC << endl;
621                 cout << "Balance: " << CustomerAccounts[i].CurrentBalance
                      << endl;
622                 cout << "-------------------------------" << endl;
623             }
624             return; // Exit function after displaying details
625         }
626         else
627         {
628             // Increment retry count and notify the employee
629             retryCount++;
630             cout << "Incorrect PIN! You have " << (3 - retryCount) << "
                  attempt(s) remaining.\n";
631 
632             if (retryCount >= 3)
633             {
634                 cout << "Too many incorrect attempts. Access denied.\n";
635                 return; // Exit function if max retries are exceeded
636             }
637         }
638     }
639     EndLine;
640 }
641 
642 bool isValidCNIC(const string& CNIC)
643 {
644     // Check length of CNIC first
645     if (CNIC.length() != 15)
646     {
647         return false;
648     }
649 
650     // Check the correct positions of dashes
```

```
651     if (CNIC[5] != '-' || CNIC[13] != '-')
652     {
653         return false;
654     }
655
656     // Check if all characters (except dashes) are digits
657     for (int i = 0; i < CNIC.length(); ++i)
658     {
659         if ((i != 5 && i != 13) && !isdigit(CNIC[i]))
660         {
661             return false;
662         }
663     }
664
665     return true;
666 }
667
668 void EndLine()
669 {
670     cout <<                                                                ↵
          "\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ↵
          ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n";
671 }
```