```cpp
#include<iostream>
#include<ctime>

using namespace std;

//Class to create date type

class Date {
    int day;
    int month;
    int year;

public:
    //Constructor

    Date() {
        time_t now = time(0);
        tm *ltm = localtime(&now);
        year = 1900 + ltm->tm_year;
        month = 1 + ltm->tm_mon;
        day = ltm->tm_mday;
    }

    friend ostream & operator<<(ostream& out, const Date& c);
};

//To display date

ostream & operator<<(ostream &ost, const Date &c) {
    ost << c.day << "-" << c.month << "-" << c.year << endl;
}

class Transaction {
    int ac_no;
    int type; //1- withdrawal 0-deposit
    Date d;
    double amount;

public:

    Transaction(int ano = 0, int t = -1, double a = 0) {
        ac_no = ano;
        type = t;
        Date dt;
        d = dt;
        amount = a;
    }

    //Function to display a transaction

    void displayTrans() {
        cout << "\n====================================" << endl;
        cout << "TRANSACTION DETAILS" << endl;
        string ty = (type == 0) ? "Deposit" : "Withdrawal";
        cout << "Account number: " << ac_no << endl << "Date: " << d << "Type: "
                << ty << endl << "Amount: " << amount << endl;
        cout << "====================================\n" << endl;
    }
};

class Account //To store account details an abstract class
{
```

```
63 protected:
64     int acno;
65     string name;
66     double balance;
67     Transaction t[100];
68     int noOfTrans;
69
70 public:
71     //Parameterized constructor
72
73     Account(int ano, string na, double bal) {
74         acno = ano;
75         name = na;
76         balance = bal;
77         noOfTrans = 0;
78     }
79     //Set balance
80
81     void setBal(double b) {
82         balance = b;
83     }
84     //Get balance
85
86     double getBal() {
87         return balance;
88     }
89
90     //Get account number
91
92     int getAcno() {
93         return acno;
94     }
95
96     //Function to display account details
97
98     void display() {
99         cout << "================================================\n" << endl;
100        cout << "YOUR ACCOUNT DETAILS ARE:" << endl << "Account number: "
101                << acno << endl << "Name: " << name << endl << "Balance: Rs."
102                << balance << " (if negative then consider overdraft)" << endl;
103        cout << "================================================\n" << endl;
104    }
105
106    //Display all transactions for an account
107
108    void displayTransactions() {
109        int i;
110        for (i = 0; i < noOfTrans; i++)
111            t[i].displayTrans();
112    }
113
114    //Function to deposit
115    virtual void deposit(double d) = 0;
116    //Function to withdraw
117    virtual void withdraw(double d) = 0;
118
119 };
120
121 //Class for savings account
122
123 class Savings: public Account {
124 public:
```

```
125     //Constructor
126
127     Savings(int ano = 0, string n = "", double bal = 0.0) :
128             Account(ano, n, bal) {
129     }
130     //Overriding deposit
131
132     void deposit(double d) {
133         Transaction temp(acno, 0, d);
134         t[noOfTrans++] = temp;
135         balance += d;
136         cout << "Deposit successful" << endl;
137     }
138     //Overriding withdraw
139
140     void withdraw(double d) {
141         if (balance <= 500 || (balance - d) < 500)
142             cout << "Cannot withdraw...minimum balance must be maintained\n";
143         else {
144
145             //Storing all transactions
146             Transaction temp(acno, 1, d);
147             t[noOfTrans++] = temp;
148
149             //Update balance
150             balance -= d;
151             cout << "Withdraw successful" << endl;
152         }
153     }
154
155 };
156
157 //Class for current account
158
159 class Current: public Account {
160 public:
161     //Constructor
162
163     Current(int ano = 0, string n = "", double bal = 0.0) :
164             Account(ano, n, bal) {
165     }
166     //Overriding deposit
167
168     void deposit(double d) {
169         Transaction temp(acno, 0, d);
170         t[noOfTrans++] = temp;
171
172         //Update balance
173         balance += d;
174         cout << "Deposit successful" << endl;
175     }
176     //Overriding withdraw
177
178     void withdraw(double d) {
179         if (balance <= -20000 || (balance - d) < -20000)
180             cout << "Cannot withdraw..overdraft limit reached" << endl;
181         else {
182             //Storing all transactions
183             Transaction temp(acno, 1, d);
184             t[noOfTrans++] = temp;
185
186             //Update balance
```

```
187             balance -= d;
188             cout << "Withdraw successful" << endl;
189         }
190     }
191
192 };
193
194 //Class for storing all the Savings account
195
196 class Savings_list {
197     Savings list[100];
198     int count;
199
200 private:
201     //Function to check whether name is valid
202
203     int isName(string n) {
204         int flag = 1, i;
205         for (i = 0; i < n.length(); i++)
206             if (n[i] != ' ')
207                 if (n[i] < 65 || (n[i] > 90 && n[i] < 97) || n[i] > 122) {
208                     flag = 0;
209                     break;
210                 }
211         return flag;
212     }
213
214 public:
215
216     Savings_list() {
217         count = 0;
218     }
219
220     //Function to get total number of current account
221
222     int getCount() {
223         return count;
224     }
225     //Function to add a savings account
226
227     void addSavings() {
228         double amount;
229         string name;
230
231         //Accept name
232         do {
233             cout << "Enter name of account holder " << endl;
234             cin >> name;
235             if (!isName(name))
236                 cout << "Name not valid" << endl;
237         } while (!isName(name));
238
239         //Accept starting balance
240         do {
241             cout << "Enter amount to deposit" << endl;
242             cin >> amount;
243             if (amount < 500)
244                 cout << "Invalid amount...at least Rs.500 must be deposited\n";
245
246         } while (amount < 500);
247
248         Savings s(10000 + count, name, amount);
```

```
249
250         cout << "Your account details are " << endl;
251         s.display();
252         list[count++] = s;
253         cout << "Account successfully added" << endl;
254     }
255
256     //Function to check whether an account number is in list
257
258     int isPresent(int ano) {
259         int i, flag = 0;
260         for (i = 0; i < count; i++) {
261             if (list[i].getAcno() == ano) {
262                 flag = 1;
263                 break;
264             }
265         }
266         if (flag == 1)
267             return i;
268         else
269             return -1;
270     }
271
272     //Get details of an account
273
274     void dispDetails(int ano) {
275         list[isPresent(ano)].display();
276     }
277
278     //For transaction
279
280     int transactionWithAcno(int ano) {
281         int type;
282         double amount;
283         do {
284             cout << "Enter type of transaction" << endl << "0. Deposit" << endl
285                     << "1. Withdrawal" << endl;
286             cin >> type;
287             if (type != 0 && type != 1)
288                 cout << "Invalid choice" << endl;
289         } while (type != 0 && type != 1);
290
291         do {
292             cout << "Enter amount" << endl;
293             cin >> amount;
294             if (amount <= 0)
295                 cout << "Invalid amount" << endl;
296
297         } while (amount <= 0);
298         if (type == 0)
299             list[isPresent(ano)].deposit(amount);
300         else
301             list[isPresent(ano)].withdraw(amount);
302         return 1;
303     }
304
305     //Function to display all transactions for a particular account
306
307     void displayAllTrans(int acno) {
308         list[isPresent(acno)].displayTransactions();
309     }
310
```

```
311 };
312
313 //Class for current account
314
315 class Current_list {
316     Current list[100];
317     int count;
318
319 private:
320     //Function to check whether name is valid
321
322     int isName(string n) {
323         int flag = 1, i;
324         for (i = 0; i < n.length(); i++)
325             if (n[i] != ' ')
326                 if (n[i] < 65 || (n[i] > 90 && n[i] < 97) || n[i] > 122) {
327                     flag = 0;
328                     break;
329                 }
330         return flag;
331     }
332
333 public:
334
335     Current_list() {
336         count = 0;
337     }
338
339     //Function to get total number of current account
340
341     int getCount() {
342         return count;
343     }
344
345     //Function to add a savings account
346
347     void addCurrent() {
348         double amount;
349         string name;
350
351         //Accept name
352         do {
353             cout << "Enter name of account holder " << endl;
354             cin >> name;
355             if (!isName(name))
356                 cout << "Name not valid" << endl;
357         } while (!isName(name));
358
359         //Accept starting balance
360         do {
361             cout << "Enter amount to deposit" << endl;
362             cin >> amount;
363             if (amount <= 0)
364                 cout << "Invalid amount...at least Rs.500 must be deposited\n";
365
366         } while (amount <= 0);
367
368         Current s(20000 + count, name, amount);
369
370         cout << "Your account details are " << endl;
371         s.display();
372         list[count++] = s;
```

```
373         cout << "Account successfully added" << endl;
374     }
375
376     //Function to check whether an account number is in list
377
378     int isPresent(int ano) {
379         int i, flag = 0;
380         for (i = 0; i < count; i++) {
381             if (list[i].getAcno() == ano) {
382                 flag = 1;
383                 break;
384             }
385         }
386         if (flag == 1)
387             return i;
388         else
389             return -1;
390     }
391
392     //Get details of an account
393
394     void dispDetails(int ano) {
395         list[isPresent(ano)].display();
396     }
397
398     //For transaction
399
400     int transactionWithAcno(int ano) {
401         int type;
402         double amount;
403         do {
404             cout << "Enter type of transaction" << endl << "0. Deposit" << endl
405                     << "1. Withdrawal" << endl;
406             cin >> type;
407             if (type != 0 && type != 1)
408                 cout << "Invalid choice" << endl;
409         } while (type != 0 && type != 1);
410
411         do {
412             cout << "Enter amount" << endl;
413             cin >> amount;
414             if (amount <= 0)
415                 cout << "Invalid amount" << endl;
416         } while (amount <= 0);
417         if (type == 0)
418             list[isPresent(ano)].deposit(amount);
419         else
420             list[isPresent(ano)].withdraw(amount);
421         return 1;
422     }
423
424     //Function to display all transactions for a particular account
425
426     void displayAllTrans(int acno) {
427         list[isPresent(acno)].displayTransactions();
428     }
429
430 };
431
432 int main() {
433     int ch, acno;
434     Savings_list listS;
```

```
435     Current_list listC;
436
437     do {
438         cout << "1. Add new savings account" << endl
439                 << "2. Add new current account" << endl
440                 << "3. Transact with existing savings account" << endl
441                 << "4. Transact with existing current account" << endl
442                 << "5. Display details of existing savings account" << endl
443                 << "6. Display details of existing current account" << endl
444                 << "7. Display transaction details of existing savings account"
445                 << endl << "8. Display transaction details of existing "
446                         "current account" << endl << "9. Exit" << endl
447                 << "Enter choice" << endl;
448         cin >> ch;
449
450         switch (ch) {
451         case 1:
452
453             listS.addSavings();
454             break;
455
456         case 2:
457
458             listC.addCurrent();
459             break;
460
461         case 3:
462
463             if (listS.getCount() == 0) {
464                 cout << "Add some new savings account first" << endl;
465                 break;
466             }
467             do {
468                 cout << "Enter account number" << endl;
469                 cin >> acno;
470                 if (listS.isPresent(acno) == -1)
471                     cout << "Invalid account number..Re-enter" << endl;
472             } while (listS.isPresent(acno) == -1);
473             listS.transactionWithAcno(acno);
474             break;
475
476         case 4:
477
478             if (listC.getCount() == 0) {
479                 cout << "Add some new current account first" << endl;
480                 break;
481             }
482             do {
483                 cout << "Enter account number" << endl;
484                 cin >> acno;
485                 if (listC.isPresent(acno) == -1)
486                     cout << "Invalid account number..Re-enter" << endl;
487             } while (listC.isPresent(acno) == -1);
488             listC.transactionWithAcno(acno);
489             break;
490
491         case 5:
492
493             if (listS.getCount() == 0) {
494                 cout << "Add some new savings account first" << endl;
495                 break;
496             }
```

```
497             do {
498                 cout << "Enter account number" << endl;
499                 cin >> acno;
500                 if (listS.isPresent(acno) == -1)
501                     cout << "Invalid account number..Re-enter" << endl;
502             } while (listS.isPresent(acno) == -1);
503             listS.dispDetails(acno);
504             break;
505 
506         case 6:
507 
508             if (listC.getCount() == 0) {
509                 cout << "Add some new current account first" << endl;
510                 break;
511             }
512             do {
513                 cout << "Enter account number" << endl;
514                 cin >> acno;
515                 if (listC.isPresent(acno) == -1)
516                     cout << "Invalid account number..Re-enter" << endl;
517             } while (listC.isPresent(acno) == -1);
518             listC.dispDetails(acno);
519             break;
520 
521         case 7:
522 
523             if (listS.getCount() == 0) {
524                 cout << "Add some new savings account first" << endl;
525                 break;
526             }
527             do {
528                 cout << "Enter account number" << endl;
529                 cin >> acno;
530                 if (listS.isPresent(acno) == -1)
531                     cout << "Invalid account number..Re-enter" << endl;
532             } while (listS.isPresent(acno) == -1);
533             listS.displayAllTrans(acno);
534             break;
535 
536         case 8:
537 
538             if (listC.getCount() == 0) {
539                 cout << "Add some new current account first" << endl;
540                 break;
541             }
542 
543             do {
544                 cout << "Enter account number" << endl;
545                 cin >> acno;
546                 if (listC.isPresent(acno) == -1)
547                     cout << "Invalid account number..Re-enter" << endl;
548             } while (listC.isPresent(acno) == -1);
549 
550             listC.displayAllTrans(acno);
551             break;
552 
553         case 9:
554             cout << "Quitting" << endl;
555             exit(0);
556 
557         default:
558             cout << "Invalid choice" << endl;
```

```
559
560         }
561     } while (ch != 9);
562     return 0;
563 }
564
```