1. Que font les programmes suivant ?
2. Décrivez en détail le code et son fonctionement.
3. Ce code peut poser problème si plusieurs processus indépendants essayent d'accéder simultanéments au mêmes comptes. Expliquez ?
4. Comment résoudre ce problème ?

initialize.c

```c
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>


int main( int argc, char **argv ) {
  char* dataFile = argv[1];
  int fd = open( dataFile, O_WRONLY|O_CREAT|O_TRUNC, 0600 );
  int accounts[] = { 100, 20, 50, 1000, 5 };
  write( fd, accounts, sizeof(int) * 5 );
  close(fd);
  return EXIT_SUCCESS;
}
```

operations.c

```c
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>

#define ACC_SIZE 5
#define SIZE sizeof(int)

void display( int fd ) {
  int i,a;
  lseek( fd, 0, SEEK_SET );
  for( i=0; i < ACC_SIZE; i++ ) {
    read( fd, &a, SIZE );
    printf( "%d: %d\n", i, a );
  }
}

int get( int fd, int acc ) {
  int off = acc * SIZE;
  int a = -1;
  lseek( fd, off, SEEK_SET );
  read( fd, &a, SIZE );
  return a;
}
```

```
27
28 void set( int fd, int acc, int total ) {
29   int off = acc * SIZE;
30   lseek( fd, off, SEEK_SET );
31   write( fd, &total, SIZE );
32 }
33
34 void acquire( int fd, int acc ) {
35   struct flock fl;
36   fl.l_type = F_WRLCK;
37   fl.l_whence = SEEK_SET;
38   fl.l_start = acc;
39   fl.l_len = 1;
40   fcntl(fd, F_SETLKW, &fl);
41 }
42
43 void transfer( int fd, int from, int to, int amount ) {
44   int fromTotal, toTotal;
45   acquire( fd, from );
46   fromTotal = get( fd, from );
47   if( fromTotal < amount ) {
48     printf("Not enough money in account: %d\n", from );
49     return;
50   }
51   acquire( fd, to );
52   toTotal = get( fd, to );
53   set( fd, from, fromTotal-amount );
54   set( fd, to, toTotal+amount );
55 }
56
57 int main( int argc, char **argv ) {
58   char* dataFile = argv[1];
59   int fd = open( dataFile, O_RDWR|O_CREAT, 0600 );
60   printf( "=== AVANT ======================\n" );
61   display( fd );
62
63
64   transfer( fd, 0, 1, 25 );
65   transfer( fd, 3, 2, 310 );
66   transfer( fd, 4, 1, 10 );
67
68   printf( "\n=== APRES ======================\n" );
69   display( fd );
70   close(fd);
71   return EXIT_SUCCESS;
72 }
```

Makefile

```
1 CC = gcc
2 CFLAGS = -g -Wall --pedantic
3
4 all: initialize.out operations.out
5
6 initialize.out: initialize.o
7         $(CC) $(CFLAGS) initialize.o -o initialize.out
```

```
8
9  operations.out: operations.o
10         $(CC) $(CFLAGS) operations.o -o operations.out
11
12 %.o: %.c
13         $(CC) $(CFLAGS) -c $<
14
15 clean:
16         rm -f *.o *.dat *.out
```