CSCI 135 Software Design and Analysis, C++
Homework 5
Solution

Saad Mneimneh
Hunter College of CUNY

Problem 1: Uncertain Bool
Consider the following incomplete class declaration:

```
class UncertainBool {
  char b;
 public:
  .
  .
  .
};
```

The idea is to extend the bool type to express values like true, false, and uncertain. All the following functions are to be implemented as part of the class.

(a) Implement a constructor that takes one character as a parameter, 'T', 'F', or '?' and initializes the uncertain bool, by setting its private data member b accordingly. If the parameter is none of these characters, it is interpreted as '?'. We will assume that the value of the uncertain bool is true if b='T', false if b='F', and uncertain if b='?'.

(b) Implement a constructor that takes one bool as parameter and initializes the uncertain bool accordingly.

(c) Implement a default constructor that initialized the undertain bool to uncertain.

(d) Implement three functions isTrue, isFlase, and isUncertain that return true if the value of the uncertain bool is true, false, and uncertain, respectively; otherwise, return false.

(e) Implement a function called print that outputs the value of the uncertain bool as "true", "false", or "uncertain".

Problem 2: A logic that "socks"
(a) Write a function called uand to compute the logical and of two uncertain bools. The function must return an uncertain bool.

(b) Write a function called uor to compute the logical or of two uncertain bools. The function must return an uncertain bool.

(c) Write a function called unot to compute the logical negation on an uncertain bool. The function must return an uncertain bool.

(d) Write a function called uequal to test whether two uncertain bools are equal. The function must return an uncertain bool.

(e) In main, have the user enter a response to the following questions:

- are your socks cotton?
- are your socks red?

The user can enter a positive number meaning yes, a zero meaning no, or a negative number meaning he does not know.

Use two uncertain bools to record the answers of the user. Then output the result of the following assertions:

- the socks are cotton but not red
- the socks are not cotton or they are red
- the socks are cotton and red, or not cotton and not red

Solution:

```
class UncertainBool {
  char b;
 public:
  UncertainBool(char c) {
    if (c=='T' || c=='F')
      b=c;
    else
      b='?';
  }

  UncertainBool(bool b) {
    if (b)
      this->b='T'; //avoid shadowing
    else
      this->b='F'; //using the implicit pointer
  }

  UncertainBool() {
    b='?';
  }

  bool isTrue() {
    return (b=='T');
  }

  bool isFalse() {
    return (b=='F');
  }

  bool isUncertain() {
    return (b=='?');
  }

  void print() {
    if (b=='T')
      cout<<"true";
    else if (b=='F')
      cout<<"false";
    else cout<<"uncertain";
  }
};
```

```
UncertainBool uand(UncertainBool b1, UncertainBool b2) {
  if (b1.isTrue() && b2.isTrue())
    return UncertainBool(true);
  else if (b1.isFalse() || b2.isFalse())
    return UncertainBool(false);
  else
    return UncertainBool();
}

UncertainBool uor(UncertainBool b1, UncertainBool b2) {
  if (b1.isTrue() || b2.isTrue())
    return UncertainBool(true);
  else if (b1.isFalse() && b2.isFalse())
    return UncertainBool(false);
  else
    return UncertainBool();
}

UncertainBool unot(UncertainBool b) {
  if (b.isUncertain())
    return b;
  else
    return UncertainBool(b.isFalse());
}

UncertainBool uequal(UncertainBool b1, UncertainBool b2) {
  if (b1.isUncertain() || b2.isUncertain())
    return UncertainBool();
  else
    return UncertainBool((b1.isTrue() && b2.isTrue()) ||
                         (b1.isFalse() && b2.isFalse()));
}
```

```
int main() {
  int r;
  int c;
  UncertainBool cotton;
  UncertainBool red;
  cout<<"are your socks cotton?";
  cin>>c;
  if (c>=0)
    cotton=UncertainBool(c); // c interpreted as bool
  cout<<"are your socks red?";
  cin>>r;
  if (r>=0)
    red=UncertainBool(r); // r interpreted as bool
  UncertainBool b;

  cout<<"the socks are cotton but not red: ";
  b=uand(cotton, unot(red));
  b.print();
  cout<<'\n';

  cout<<"the socks are not cotton or they are red: ";
  b=uor(unot(cotton), red);
  b.print();
  cout<<'\n';

  cout<<"the socks are cotton and red, or not cotton and not red: ";
  b=uequal(cotton, red);
  b.print();
  cout<<'\n';
};
```

Problem 3: Jumping game revisited
Refer to the previous homework. This time, an array of length n contains integers less than n, but some may be negative. A negative entry means its missing. Therefore, if the jumping game reaches a missing entry, the result of the game is not known. Repeat the previous homework, but the function terminate should now return an uncertain bool.

Solution:

```
UncertainBool terminate(int * a, int n) {
  int i=a[0];
  for (int step=1; step<n && i>0; step=step+1) //this will stop after n-1
    i=a[i];  //jump                           //steps or when i<=0
  if (i==0)
    return UncertainBool(true);
  else if (i>0)  //too many steps
    return UncertainBool(false);
  else  //i<0
    return UncertainBool();
}


UncertainBool terminate(int * a) {
  int i=a[0];
  while (i>0) {  //this will stop if i<=0
    if (a[i]==i) //loop
      return UncertainBool(false);
    int j=i;
    i=a[i];  //jump
    a[j]=j;  //a[j] visited, create a loop at a[j]
  }
  if (i==0)
    return UncertainBool(true);
  else  //i<0
    return UncertainBool();
}
```