3. Oletetaan, että tarvittavat include-direktiivit yms. ovat käytössä. Mitä seuraava ohjelma tulostaa? (2p.)

```
bool foo( int f ) {
   f = f + 1;
   cout << "foo " << f << endl;
   if( f < 3 ) {
      return true;
   }
   return false;
}

int bar( int& b ) {
   b = b + 1;
   cout << "bar " << b << endl;
   return b;
}

int main() {
   int luku = 0;
   while( foo( bar ( luku ) ) ) {
      cout << "main " << luku << endl;
   }
   return EXIT_SUCCESS;
}
```

#include <string>

Jäsenfunktiot

char at(*int*)	Merkkijonon indeksointi kohdasta *int*. Voi olla sijoitusoperaattorin vasemmalla puolella.
string::size_type length()	Merkkijonon pituus.
bool empty()	true, jos merkkijono on tyhjä, false jos ei.
void clear()	Tyhjentää merkkijonon.
string erase()	Tyhjentää merkkijonon.
string erase(*int*)	Tuhoaa kaikki merkit alkaen indeksistä *int*.
string erase(int_1, int_2)	Tuhoaa int_2 merkkiä alkaen merkistä int_1.
string append(*string*)	Liittää merkkijonon *string* merkit loppuun.
string insert(*int*, *string*)	Lisää merkkijonon *string* merkit kohtaan *int*.
string replace(int_1, int_2, *string*)	Korvaa kohdasta int_1 alkaen int_2 merkkiä merkkijonolla *string*.
string substr(*int*)	Palauttaa lopun merkkijonon alkaen indeksistä *int*.
string substr(int_1, int_2)	Palauttaa indeksistä int_1 alkavan int_2 pituisen alimerkkijonon.
const char* c_str()	Konversio C-merkkijonoksi.
string::size_type find(*char*, *int* = 0)	
string::size_type find(*string*, *int* = 0)	Etsitään merkkiä *char* tai alimerkkijonoa *string* alkaen indeksistä *int*. Paluuarvo löytyneen kohdan indeksi tai string::npos, jos ei löytynyt.
string::size_type rfind(...)	Kuten find edellä, mutta etsintä tapahtuu merkkijonon lopusta alkua kohti.

Tehtävä 4

1. Minkälainen muuttuja seuraavassa määritellään? `int luvut[ KOKO ];` (1p.)
2. Oletetaan, että tarvittavat include-direktiivit, muuttujien määrittelyt jne. ovat kunnossa. Selitä yksinkertaisesti suomeksi, mitä seuraava ohjelmapätkä tekee. (2p.)

```
bool tarkastus = true;
for( unsigned int i = 0; i < KOKO - 1; ++i ) {
    if( luvut[ i ] > luvut[ i + 1 ] ) {
        tarkastus = false;
    }
}
```

3. Muuta 2. kohdan ohjelmapätkää niin, että `luvut` käydään läpi lopusta alkuun päin, mutta ohjelma toimii silti samalla tavalla kuin alkuperäinen. (2p.)
4. Miksi 2. kohdassa esitetyssä ohjelmapätkässä `for`-silmukan ehdossa verrataan `i`:n arvoa arvoon `KOKO - 1` eikä arvoon `KOKO`? (1p.)
5. Kirjoita ohjelmapätkä, joka tarkastaa, onko taulukon alkioiden järjestys sama alusta loppuun ja lopusta alkuun lueteltaessa. Esim. 1, 2, 1 ja 9, 7, 7, 9 ovat, mutta 6, 7, 9, 6 ei ole. (4p.)
6. Arvioi, onko kohtaan 5. kirjoittamasi ohjelmapätkä toteutettu fiksusti. Voisiko saman tominnallisuuden toteuttaa jollain toisella tavalla? Olisiko tämä toinen tapa parempi vai huonompi kuin toteuttamasi ohjelma? Perustele. (2p.)