Rešitve pisnega izpita (programska jezika Java in C) z dne 17. junija 2005

1. naloga (25%)

```
#include <stdio.h>

int main(int argc, char *argv[]) {
  FILE *fp1, *fp2;
  int zn1, zn2;
  int vrstice = 1;
  int znaki = 1;

  fp1 = fopen(argv[1], "r");
  fp2 = fopen(argv[2], "r");
  while( ((zn1=getc(fp1)) != EOF) && ((zn2=getc(fp2)) != EOF) && (zn1 == zn2) ) {
    znaki++;
    if(zn1 == '\n') {
      vrstice++;
      znaki = 1;
    }
  }
  if (zn1 == EOF) {
    zn2 = getc(fp2);
    if( zn1 != zn2 )
      printf("Datoteki se razlikujeta v vrstici: %d, v znaku: %d\n", vrstice, znaki);
    else
      printf("Datoteki sta enaki.\n");
  }
  else
    printf("Datoteki se razlikujeta v vrstici: %d, v znaku: %d\n", vrstice, znaki);
  fclose(fp1);
  fclose(fp2);
  return(0);
}
```

2. naloga (25%)

Recimo, da je naša vpisna številka enaka 63040789.

Program izpiše naslednje:

```
63040789
63040789
63040879
```

Pri tem se izvedejo naslednji klici funkcije *abc*:

```
abc("63040789",0)
abc("63040789",1)
abc("63040789",2)
abc("63040879",2)
```

Obrazložitev:
Funkcija *abc* prejme dva argumenta, prvi je niz znakov, drugi pa število *n*, ki je ob prvem klicu funkcije enako nič. Funkcija dela permutacije znakov podanega niza od petega znaka naprej, *n* pa določa število znakov, nad katerimi naredi zamenjave. Pri $n = 2$ prenehamo z zamenjavami in se niz znakov le izpiše.
Ob prvem klicu funkcije *abc* je $n = 0$, zato se izvede le ponoven klic funkcije *abc* z $n = 1$, ne pa tudi for zanka, saj pogoj za ponovitev zanke $i < n$ že na začetku ni izpolnjen (0 ni strogo manjše od 0).
Ob klicu funkcije *abc* pri $n = 1$ se najprej izvede ponoven klic funkcije *abc* z $n = 2$, nato pa še for zanka, ki se izvede le enkrat (za $i = 0$, medtem ko za $i = 1$ pogoj $i<n$ ni več izpolnjen). Znotraj for zanke se najprej zamenjata 6. in 7. znak podanega niza (znaka z indeksoma 5 in 6), nato se kliče funkcija *abc* z $n = 2$ in s spremenjenim nizom, na koncu pa se ponovno zamenjata 6. in 7. znak niza, tako da spet dobimo nazaj podani niz.
Ob klicu funkcije *abc* pri $n = 2$ pa se, kot smo že rekli, le izpiše podani niz.

3. naloga (25%)

Rekurzivna rešitev:

```
// najkrajsa je rekurzivna resitev
int obrni(int n)
{
  if (n/10 == 0)
    return n;
  return Integer.parseInt(Integer.toString(n%10) + Integer.toString(obrni(n/10)));
}
```

Iterativna rešitev z uporabo objekta razreda StringBuffer:

```
// iterativna resitev, pomagamo si z razredom StringBuffer
int obrni(int n)
{
  StringBuffer niz = new StringBuffer(Integer.toString(n));
  for(int i=0; i<niz.length()/2; i++) {
    char znak = niz.charAt(i);
    niz.setCharAt(i, niz.charAt(niz.length()-1-i));
    niz.setCharAt(niz.length()-1-i, znak);
  }
  return Integer.parseInt(niz.toString());
}
```

ali krajše z uporabo metode *reverse*:

```
int obrni(int n)
{
  StringBuffer niz = new StringBuffer(Integer.toString(n));
  niz.reverse();
  return Integer.parseInt(niz.toString());
}
```

4. naloga (25%)

```
public class KomplStevilo
{
  private double realni;
  private double imaginarni;

  public KomplStevilo(double r, double i)
  {
    this.realni = r;
    this.imaginarni = i;
  }

  public void pristej(KomplStevilo stev)
  {
    this.realni += stev.vrniRealni();
    this.imaginarni += stev.vrniImaginarni();
  }

  public KomplStevilo duplikat()
  {
    return new KomplStevilo(this.realni, this.imaginarni);
  }

  public double vrniRealni()
  {
    return this.realni;
  }

  public double vrniImaginarni()
  {
    return this.imaginarni;
  }
}
```