Structuur van computerprogramma's II

1. Een *palindroom* is een eindige rij van elementen $e_1 e_2 \ldots e_n$, $n \geq 0$ die zodanig is dat $e_1 e_2 \ldots e_n = e_n e_{n-1} \ldots e_1$, m.a.w. de rij van links naar rechts of van rechts naar links "lezen" geeft hetzelfde resultaat. Schrijf een zo kort mogelijke generische functie die nagaat of een rij een palindroom is. De functie moet in elk geval werken voor std :: string objecten. (Hint: iterators).

2. Is de volgende code (semantisch) correct? Indien niet: corrigeer.

```
1  #include <iostream>
2  #include <sstream>
3  #include <string>
4  #include <set>
5
6  template<typename T>
7  std :: string
8   tostring (const T& t) {
9     std :: ostringstream  oss;
10    oss << t;
11    return oss.str();
12 }
13
14 class A {
15   public:
16     A(int a=3): a_(a) {}
17     std :: string  str ()  const { return tostring(a_); }
18   private:
19     int a_;
20 };
21
22 int
23 main(int,char**) {
24    std :: set<A> a_set;
```

```
25    for ( size_t  i=0;  i<10; ++i)
26      a_set . insert (A(i ));
27    for (std :: set<A>::iterator  i=a_set . begin ();  i!= a_set .end ();  ++i)
28      std :: cout  << i->str() << std::endl;
29    return 0;
30  }
```

3. Compileert het onderstaande programma? Indien ja: wat is de output. Indien neen: waarom niet?

```
1  #include <iostream>
2
3  class A {
4    public:
5      A(int i): data(i) { std :: cout << "A::A(" << i << ")" << std::endl; }
6      A(const char* s): data(0) { std :: cout << "A::A(" << s << ")" << std::endl; }
7      ~A() { std :: cout << "A::~A()" << std::endl; }
8    private:
9      int data;
10 };
11
12 class B: public A {
13   public:
14     B(const A& a): A(a) { std :: cout << "B::B(const A& a)" << std::endl; }
15     ~B() { std :: cout << "B::~B()" << std::endl; }
16 };
17
18 int
19 main(int, char**) {
20   B b0(1);
21   B b1("abc");
22   return 0;
23 }
```

4. Bespreek in detail de motivatie (waarom wordt wat gebruikt) van de verschillende constructies in de volgende code. Hoe zou base_type nog nuttig kunnen uitgebreid worden?

```
1  #include <string>
2  #include <sstream>
3  #include <iostream>
4
```

```
5  template <typename T>
6    struct base_type {
7      typedef T mutable_type;
8    };
9
10 template <typename T>
11   struct base_type<const T> {
12     typedef T mutable_type;
13   };
14
15 template<class T>
16   T parse(const std:: string & s) {
17     typename base_type<T>::mutable_type t;
18     std :: istringstream iss(s);
19     iss >> t;
20     return t;
21   }
22
23 int
24 main(int,char**) {
25   const int x(parse<const int>("3"));
26   std :: cout << x << std::endl;
27   return 0;
28 }
```