Musterlösung zur

Beispiel-Klausur Wintersemester 2004 / 05

Aufgabe 1 (15 Punkte)

```
#include <math.h>

class Vektor2D {
private:
    float x, y;
public:
    Vektor2D(float v_x, float v_y);

    void  print();    // Ausgabe des Vektors auf der Konsole in der Form "x | y"
    float abs();      // Laenge des Vektors

    static Vektor2D   add(Vektor2D a, Vektor2D b);   // Vektoraddition von a und b
    static Vektor2D   sub(Vektor2D a, Vektor2D b);   // Vektorsubtraktion von a und b
    static float    ska_p(Vektor2D a, Vektor2D b);   // Skalarprodukt der Vektoren a und b
    static float    angle_d(Vektor2D a, Vektor2D b); // Winkel zw. den Vektoren a und b in Grad
};


float Vektor2D::angle_d(Vektor2D a, Vektor2D b)
{
   return 180 * acos( ska_p(a,b) / (a.abs()*b.abs()) ) / M_PI;
}
```

Aufgabe 2 (20 Punkte)

```
template<class T>
void arrayprint(T* a, int n)
{
   int i;
   for(i=0; i<n-1; i++) {
      cout << a[i] << ", ";
   }
   cout << a[i];
}
```

Aufagbe 3 (30 Punkte)

a) (15 Punkte)

```
class pattern2 : public circle, public rectangle
{
    private:
    public:
        pattern2(char* n, color c, bool v, float pos_x, float pos_y, float radius);
        ~pattern2();

        float calc_surface();
        void display();
        void get_points(float*);
};
```

b) (15 Punkte)

```
pattern2::pattern2(char* n, color c, bool v, float pos_x, float pos_y, float radius)
   : circle(n, c, v, pos_x, pos_y, radius), rectangle(n, c, false, 0,0,0,0)
{
    float t = 0.7071 * radius;
    rectangle::ax = pos_x - t;
    rectangle::ay = pos_y - t;
    rectangle::bx = pos_x + t;
    rectangle::by = pos_y + t;
}
```

Aufgabe 4 (15 Punkte)

Statische Erzeugung :

```
animal cat("Miezi", 4);
animal dog("Waldi", 4);
```

Dynamische Erzeugung :

```
      animal            *p_horse, *p_cow;

      p_horse = new animal("Britta", 4);
      p_cow = new animal("Tilda", 4);

      p_horse->write();
      p_cow->write();
```

Aufgabe 5 (10 Punkte)

```
class queue{
      private:
            char *name;
            int limit;
            float *elements;
      public:
            queue(char *my_name, int max_elements);
            void Push(float new_element);
            float Pop();
            ~queue();
      }

queue::queue(char *my_name, int max_elements = 50) {
      name = my_name;
      limit = max_elements;
}
```