

Probeklausur

(9.1) [0 Punkte]

Was wird ausgegeben?

```
#include <iostream>
#include <ostream>
#include <curses.h>

using namespace std;

int test(int x)
{
    if (x == 0) return 1;
    int z = 0;
    for (int i=0; i<x; i++)
    {
        z = z + test(x-1);
    }
    return z;
}

int main()
{
    int z = 0;
    for (int i= 0; i < 10; i++)
    {
        if (i%3 == 1) 1,4,7
        {
            z= z + i;
        }
    }
    cout << z << "\n"; "12"
    cout << test(7);    "5040"
    return 0;
}
```

(9.2) [0 Punkte]

Schreiben Sie eine Funktion, die das kleinste Element einer verketteten Liste zurückgibt. Der Funktion wird nur der Anker der verketteten Liste übergeben.

```
struct TListenKnoten
{
    int data;
    TListenKnoten *next;
};

int minder( TListKnoten *anker){
    TListKnoten *akt=anker; int min=akt->data;
    while(anker->next!=0){
        akt=anker->next;
        if(akt->data<min)min=akt->data;
    }
    return min;
}
```

```

int vokale(string str){
    int zahl=0;
    for(int i;i<str.length;i++){
        if(str[i]=='A' || str[i]=='a' || ...) zahl++;
    }
    return zahl
}

```

(9.3) [0 Punkte]

Schreiben Sie eine Funktion, die eine Zeichenkette übergeben bekommt und die Anzahl der Vokale der Zeichenkette zurückgibt.

(9.4) [0 Punkte]

Schreiben Sie ein Programm, dass 10 Integer Zahlen vom Benutzer einliest und in ein Array speichert. Danach soll das Programm alle geraden Zahlen aus dem Array ausgeben.

(9.5) [0 Punkte]

Was wird ausgegeben?

```

#include <iostream>
#include <ostream>
#include <urses.h>
using namespace std;

```

```

class Tier
{
protected:
    string farbe; int groesse;
public:
    Tier()
    {
        cout << "Ich bin ein Tier\n";
        farbe = "blau"; groesse =50;
    }

    Tier(string Farbe)
    {
        cout << "Ich bin ein grosses Tier\n";
        farbe = Farbe; groesse =100;
    }

    void zeigeFarbe()
    {
        cout << farbe << "\n";
    }

    void virtual zeigeGroesse()
    {
        cout << groesse << "\n";
    }
};

```

```

class Hase : public Tier
{
public:
    Hase()

```

```

int main(){
    int c[10];
    for(int i=0;i<10;i++){
        cin>>c[i];
    }
    for(int i=0;i<10;i++){
        if(c[i]%2==0)cout<<c[i]<<"\n";
    }
}

```

```

{
    farbe = "braun"; groesse = 30;
    cout << "Ich bin ein Hase\n";
}

void zeigeFarbe()
{
    cout << "Hase: " << farbe << "\n";
}

void zeigeGroesse()
{
    cout << "Hase: " << groesse << "\n";
}
};

class Katze : public Tier
{
public:
    Katze()
    {
        farbe = "grau"; groesse = 20;
        cout << "Ich bin eine Katze\n";
    }
    Katze(string Farbe, int Groesse) : Tier(Farbe)
    {
        farbe=Farbe; groesse = Groesse;
    }

    void zeigeFarbe()
    {
        cout << "Katze: " << farbe << "\n";
    }

    void zeigeGroesse()
    {
        cout << "Katze: " << groesse << "\n";
    }
};

int main()
{
    Tier *t[3];
    t[0]= new Tier;    Ich bin ein Tier
    t[1]= new Hase;    Ich bin ein Tier\n ich bin ein Hase
    t[2]= new Katze("weiss", 60); Ich bin ein grosses Tier\n
    for(int i = 0; i<3; i++)
    {
        t[i]->zeigeFarbe();    0:blau\n50
        t[i]->zeigeGroesse(); 1:braun\nHase: 30
                                2:weiss\nKatze: 60
    }
    Hase h; Ich bin ein Tier\n ich bin ein Hase
    h.zeigeFarbe();Hase: braun

```

```
h.zeigeGroesse();  
Katze k;    Ich bin ein Tier\n ich bin eine Katze  
k.zeigeFarbe(); Katze: grau  
k.zeigeGroesse(); Katze: 20  
return 0;  
}
```