Code-Rätsel

Alle Codeausschnitte stammen aus den Einsendungen der ersten Runde 2015.

Codeausschnitt 1

Was macht dieser Code? Wie kann man ihn schöner und schneller machen?

```
for (int k=2; k<=n; k++){
    for (int m=1; m<=k-1; m++) {
        if (a[k].size()<a[m].size()){
            c=a[k];
            a[k]=a[m];
            a[m]=c;
        }
    }
}
```

Codeausschnitt 2

Was macht dieser Code? Wie kann man ihn schöner und schneller machen?

```
int suche(long int o,vector<long int>e1)
{
    for(int x(0);x<e1.size();x++)
    {
        if(e1[x]==o)
            return x;
    }
}
```

Codeausschnitt 3

Was macht dieser Code? Wie kann man ihn schöner und schneller machen?

```
int ifin(char * a,char * b){
        int z=0;
        if(strlen(a)>strlen(b))swap(a,b);
        for(int i=0;i<strlen(b);i++){
                if (b[i]==a[z])z++;
                if (z==strlen(a))return 1;
        }
        return 0;
}
```

Codeausschnitt 4

Was ist seltsam?

```
max=0;
for(int j=0;j<m;j++){
        scanf("%d",&n[j]);
        max+=n[j];
}
```

Codeausschnitt 5

Wie kann man diesen Code schöner machen?

```
for (int j = 0; j < q; j++)
  {
    for (; k < p && l == 0; k++)
      {
        if (B[j] == A[k])
          {
            l = 1;
            counter += 1;
          }
      }
    l = 0;
  }
```

Codeausschnitt 6

Verkürze auf 4 Zeilen:

```
if(p+1 == jurte[p]){
}

else{
        found = false;
        int search = p;
        while(found == false){
                search++;
                if(p+1 == jurte[search]){
                        int a=jurte[p];
                        jurte[p]=jurte[search];
                        jurte[search]=a;

                        cout << " " << p+1 << " " << search+1;
                        found = true;
                }
        }
}
```

Codeausschnitt 7

Wie Code-Duplikation vermeiden?

```
for (int k=1; k<=m; k++){
    if ((a[k]==rest[ind]) && (c[b[k]]==0)){
        c[b[k]]=c[rest[ind]]%2+1;
        v+=1;
        rest[v]=b[k];
        }
    if ((b[k]==rest[ind]) && (c[a[k]]==0)){
        c[a[k]]=c[rest[ind]]%2+1;
        v+=1;
        rest[v]=a[k];
    }
}
```

Codeausschnitt 8

Verkürze auf 6 Zeilen:

```
int p=0;
while(true){
  if(y[p]==p){
    ++p;
    if(p>=n){
      break;
    }
    continue;
  }
  cout<<p+1<<" "<<y[p]+1<<" ";
  tmp = y[y[p]];
  y[y[p]] = y[p];
  y[p] = tmp;
}
```

Codeausschnitt 9

Verkürze auf 6 Zeilen:

```
int t, n1, n2;
cin>>t;
int result[t];

for(int x=0;x<t;x++){
        cin>>n1>>n2;

        if((n1+n2)/2>=n1 && (n1+n2)/2>=n2){
                result[x] = 1;
        }else{
                result[x] = 0;
        }
}

for(int x=0;x<t;x++){
        if(result[x]==1){
                cout<<"Case #"<< x+1 <<": POSSIBLE\n";
        }else{
                cout<<"Case #"<< x+1 <<": IMPOSSIBLE\n";
        }
}
```

Codeausschnitt 10

Verwende die Range-based-`for`-Schleife und verkürze auf 7 Zeilen:

```
set<int>::iterator it = times.begin();
int start=-1;
int count = 0;
for(unsigned int j=1;j<times.size();j++){
    for(unsigned int k=0;k < schwings.size();k++){
        if(schwings[k].isPlaying(*it)){
            count++;
            if(start==-1){
                start=*it;
            }
            break;
        }
    }
    it++;
}
cout << count << " " << start << endl;
```

Codeausschnitt 11

Schreibe lesbar:

```
side.at((*j).first-1) == -1?
  side.at((*j).second-1) == -1?
  side.at((*j).first-1) = Left,
  side.at((*j).second-1) = Right:
  side.at((*j).second-1)?
  side.at((*j).first-1) = Left:
  side.at((*j).first-1) = Right:
  side.at((*j).first-1)?side.at((*j).second-1) = Left:
  side.at((*j).second-1) = Right;
```

Hinweis:

```
#define Right 1
#define Left 0
```

Tipp: 4 Zeilen ohne ternärer Operator möglich

Codeausschnitt 12

Verkürze!

```
// void shortenFile(vector<Wave> &waves){
sort(waves.begin(), waves.end(), isSooner);
vector<Wave> file;
file.push_back(waves[0]);
size_t lastIndex = 0;
// merge all waves together
for (size_t wi = 1; wi < waves.size(); wi++){
bool inserted = false;
for (size_t fi = lastIndex; fi < file.size(); fi++){
        if (waves[wi].start == file[fi].start){
                if (waves[wi].end == file[fi].end){
                        file[fi].merge(waves[wi]);
                        inserted = true;
                        break;
                }
                else if(waves[wi].end > file[fi].end){
                        file[fi].merge(waves[wi]);
                        waves[wi].start = file[fi].end;
                }
                else{
                        waves[wi].merge(file[fi]);
                        file[fi].start = waves[wi].end;
                        file.insert(file.begin() + fi, waves[wi]);
                        inserted = true;
                        break;
                }
        }
        else if (waves[wi].start < file[fi].start){
                if (waves[wi].end == file[fi].end){
                        file[fi].merge(waves[wi]);
                        waves[wi].end = file[fi].start;
                        file.insert(file.begin() + fi, file[fi]);
                        inserted = true;
                        break;
                }
                else if (waves[wi].end > file[fi].end){
                        file[fi].merge(waves[wi]);
                        Wave b = waves[wi].split(file[fi].start);
                        waves[wi].start = file[fi].end;
                        file.insert(file.begin() + fi, b);
                }
                else{
                        Wave ob = file[fi].split(waves[wi].end);
                        ob.merge(waves[wi]);
                        file.insert(file.begin() + fi, ob);
                        waves[wi].end = ob.start;
                        file.insert(file.begin() + fi, waves[wi]);
                        inserted = true;
                        break;
                }
        }
        else {
                if (waves[wi].end == file[fi].end){
                        Wave ob = file[fi].split(waves[wi].start);
                        file[fi].merge(waves[wi]);
                        file.insert(file.begin() + fi, ob);
                        inserted = true;
                        break;
                }
                else if (waves[wi].end > file[fi].end){
                        Wave ob = file[fi].split(waves[wi].start);
                        file[fi].merge(waves[wi]);
                        file.insert(file.begin() + fi, ob);
                        waves[wi].start = file[fi].end;
                }
                else{
                        waves[wi].merge(file[fi]);
                        Wave ob = file[fi].split(waves[wi].end);
                        ob.end = waves[wi].start;
                        file.insert(file.begin() + fi, waves[wi]);
                        file.insert(file.begin() + fi, ob);
                        inserted = true;
                        break;
                }
        }
}
if (!inserted){
        file.push_back(waves[wi]);
}
}
```