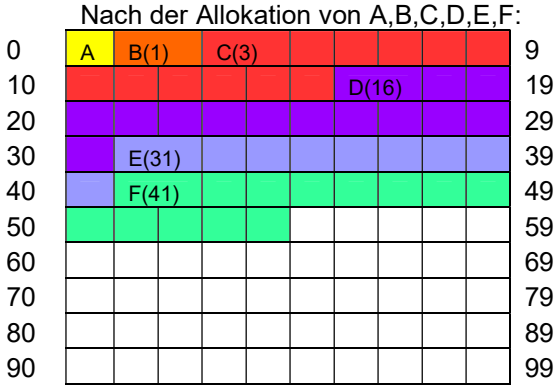
Maksym Zinnatov, 3148464

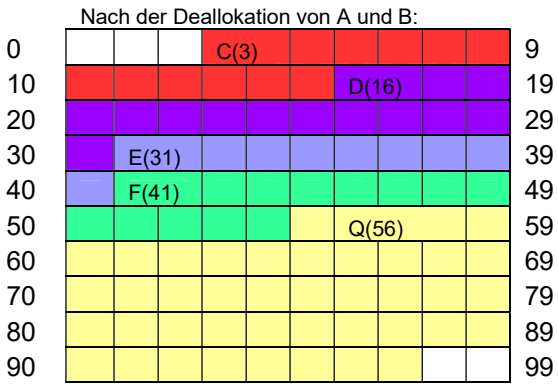
Aufgabe 3.1

Die Speicherposition steht in Klammen.

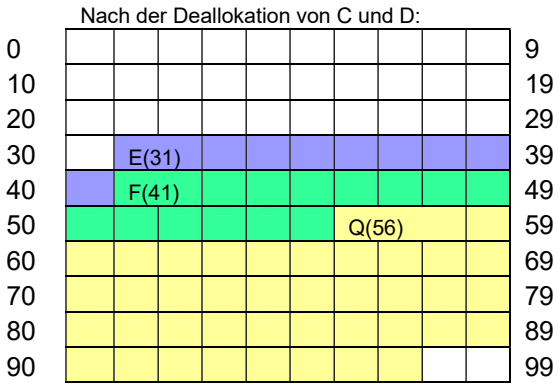
Nach der Allokation von A,B,C,D,E,F:



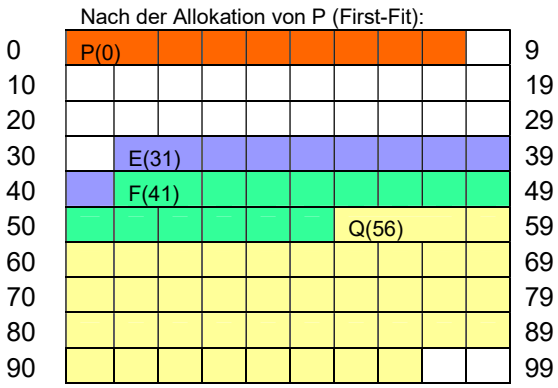
Nach der Deallokation von A und B:



Nach der Deallokation von C und D:



Nach der Allokation von P (First-Fit):



Da der freie Block in den Positionen 98-99 für P zu klein ist, wird bei der Next-Fit-Allokation der Block ***P*** ab Position 0 allokiert (also genau wie bei First-Fit). Bei Modulo-Addressierung würde er auf Positionen 98-99 und 0-6 allokiert, aber ich konnte nicht feststellen, ob das im Hauptspeicher passiert.

Maksym Zinnatov, 3148464

Aufgabe 3.5

1. Call-by-Reference

Ausgabe:
X=2 Y=5 Z=4
A=(2 1 1 4 3)

Prozedur F wird mit Parameter (X,A(Y)) aufgerufen.
F setzt X=2, Y=5, und ruft die Prozedur G mit Parameter (Z,Z_f)
G verändert Z und Z_f, sowie das 2. Element der Liste:
A(Z) = A(A(Z_f)) = 1; dann Z=2, Z_f=4,
F verändert A(X): A(X) = A(Z_f), also A(2) = A(4) = 4

Aufgabe 3.6

1. Fehler, smart_pointers.adb, Zeile 18, `function Is_Null`:

```
return X.Reference_Count = 0;
```

Änderung

```
return X=null or else X.Reference_Count = 0;
```

Testfall:

```
Assert (Is_Null (Null_Pointer));
```

Ausgabe (ohne Korrektur):

```
raised CONSTRAINT_ERROR : smart_pointers.adb:18 access check
failed
```

Ausgabe (mit Korrektur):

```
Good
```

Begründung:
Die Funktion `Is_Null` versuchte einen nichtexistierenden Objekt zuzugreifen.

2. Fehler, smart_pointers.adb, Zeile 8, `procedure Create`:

```
X := new Info'(0, Datums.Allocate);
```

Änderung:

```
X := new Info'(1, Datums.Allocate);
```

Testfall:

```
Smart_Pointers.Create (A);
Assert (not Is_Null (A));
```

Ausgabe (ohne Korrektur):

```
Allocate object 1
FAILED
```

Ausgabe (mit Korrektur):

```
Allocate object 1
Good
```

Begründung:
Ein Objekt wurde mit 0 Referenzen darauf erzeugt und gelte deshalb als Null, obwohl es einen Smart Pointer auf dieses Objekt gab.

3. Fehler, smart_pointers.adb, Zeile 52, `procedure Release:`

```
Dec (X);
```

Änderung:

```
(Zeile löschen)
```

Testfall:

```
Smart_Pointers.Create (A);
Smart_Pointers.Assign (A, B);
Smart_Pointers.Release (B);
Smart_Pointers.Print (A);
```

Ausgabe (ohne Korrektur):

```
Allocate object 1
Free object 1
ERROR: Printing deallocated object 1
```

Ausgabe (mit Korrektur):

```
Allocate object 1
Printing object 1
```

Begründung:

Es gab doppelte Dekrementierung (einmal in `procedure Release` und einmal in `procedure Assign`), deshalb wurde das Datum gelöscht.