4.1 Data structures

(a) Gegeben sei der folgende Hexdump, der eine Datenstruktur im Speicher darstellt. Wie könnte die dazugehörige Definition (z.B. in der Programmiersprache C) ausgesehen haben?

```
084c6008|4a 4b 20 52 6f 77 6c 69 6e 67 00 00 00 00 00 00|JK Rowling......|
084c6018|00 00 00 00 48 61 72 72 79 20 50 6f 74 74 65 72|....Harry Potter|
084c6028|20 61 6e 64 20 74 68 65 20 53 6f 72 63 65 72 65| and the Sorcere|
084c6038|72 27 73 20 53 74 6f 6e 65 00 00 00 30 34 33 39|r's Stone...0439|
084c6048|37 30 38 31 38 34 00 00 03 00 00 00            |708184......    |
```

4.2 Static Analysis

(a) Im Vergleich von statischem und dynamischem Linken: Welche Schwierigkeit bringt das statische Linken hinsichtlich des Reverse Engineerings?

(b) Nennen Sie mindestens 2 Schwierigkeiten beim Disassembling von x86-Maschinencode.

(c) Nennen Sie die beiden Verfahren zum Disassembling und beschreiben Sie sie.

(d) x86-Instruktionen haben eine variable Länge zwischen 1 und 15 Bytes. Je nachdem, an welcher Position innerhalb eines Bytestroms die Disassembly beginnt, ergeben sich unterschiedliche Bedeutungen. Konstruieren Sie ein Beispiel, in dem Teile einer Multibyte-Instruktion eine eigenständige Instruktion darstellen.

(e) Nennen Sie mindestens 2 Schwierigkeiten beim Dekompilieren von x86-Maschinencode.

4.3 Disassembly 1

Betrachten Sie die folgende Disassembly.

```
0804840b <func>:
 804840b:   55                      push   ebp
 804840c:   89 e5                   mov    ebp,esp
 804840e:   83 ec 10                sub    esp,0x10
 8048411:   c7 45 fc 00 00 00 00    mov    DWORD PTR [ebp-0x4],0x0
 8048418:   eb 04                   jmp    804841e <func+0x13>
 804841a:   83 45 fc 01             add    DWORD PTR [ebp-0x4],0x1
 804841e:   8b 55 fc                mov    edx,DWORD PTR [ebp-0x4]
 8048421:   8b 45 08                mov    eax,DWORD PTR [ebp+0x8]
 8048424:   01 d0                   add    eax,edx
 8048426:   0f b6 00                movzx  eax,BYTE PTR [eax]
 8048429:   84 c0                   test   al,al
 804842b:   75 ed                   jne    804841a <func+0xf>
 804842d:   8b 45 fc                mov    eax,DWORD PTR [ebp-0x4]
 8048430:   c9                      leave
 8048431:   c3                      ret
```

(a) Was macht diese Funktion? Welchen Ergebniswert gibt sie zurück?

(b) Wie viele Argumente erwartet die Funktion und wie werden sie an die Funktion übergeben?

(c) Wie viele lokale Variablen werden innerhalb der Funktion verwendet? Wo liegen sie bzw. wie werden sie im Code adressiert?

4.4 Disassembly 2

Betrachten Sie die folgende Disassembly.

```
0804842b <func>:
 804842b:   8b 54 24 04             mov    edx,DWORD PTR [esp+0x4]
 804842f:   80 3a 00                cmp    BYTE PTR [edx],0x0
 8048432:   74 10                   je     8048444 <func+0x19>
 8048434:   b8 00 00 00 00          mov    eax,0x0
 8048439:   83 c0 01                add    eax,0x1
 804843c:   80 3c 02 00             cmp    BYTE PTR [edx+eax],0x0
 8048440:   75 f7                   jne    8048439 <func+0xe>
 8048442:   f3 c3                   repz ret
 8048444:   b8 00 00 00 00          mov    eax,0x0
 8048449:   c3                      ret
```

(a) Was macht diese Funktion? Welchen Ergebniswert gibt sie zurück?

(b) Wie viele Argumente erwartet die Funktion und wie werden sie an die Funktion übergeben?

(c) Wie viele lokale Variablen werden innerhalb der Funktion verwendet? Wo liegen sie bzw. wie werden sie im Code adressiert?

(d) Was fällt im Vergleich zu bisherigen Funktionen, die Sie sich angeschaut haben auf?

4.5 Disassembly 3

Betrachten Sie die folgende Disassembly.

```
08048410 <func>:
 8048410:   b8 ff ff ff ff          mov    eax,0xffffffff
 8048415:   8b 4c 24 04             mov    ecx,DWORD PTR [esp+0x4]
 8048419:   0f 1f 80 00 00 00 00    nop    DWORD PTR [eax+0x0]     ; Multibyte-NOP
 8048420:   80 7c 01 01 00          cmp    BYTE PTR [ecx+eax+0x1],0x0
 8048425:   8d 40 01                lea    eax,[eax+0x1]
 8048428:   75 f6                   jne    8048420 <func+0x10>
 804842a:   c3                      ret

08048430 <main>:
 8048430:   83 ec 0c                sub    esp,0xc
 8048433:   c7 04 24 e0 84 04 08    mov    DWORD PTR [esp],0x80484e0
 804843a:   e8 d1 ff ff ff          call   8048410 <func>
 ...

Hex dump of section '.rodata':
  0x080484d8 03000000 01000200 61626364 65666768 ........abcdefgh
  0x080484e8 696a6b6c 6d6e6f70 71727374 75767778 ijklmnopqrstuvwx
  0x080484f8 797a0025 640a00                     yz.%d..
```

(a) Was macht die Funktion func? Welchen Ergebniswert gibt sie zurück?

(b) Wie viele Argumente erwartet die Funktion und wie werden sie an die Funktion übergeben?

(c) Wie viele lokale Variablen werden innerhalb der Funktion verwendet? Wo liegen sie bzw. wie werden sie im Code adressiert?

(d) Was fällt im Vergleich zu bisherigen Funktionen, die Sie sich angeschaut haben auf?