Grundlagen der Praktischen Informatik **Sommersemester 2023**
Dr. Henrik Brosenne Georg-August-Universität Göttingen
Institut für Informatik

Übungsblatt 09

E-Learning

Absolvieren Sie die Tests bis Di., 27.06., 8 Uhr

Die Tests sind in der Stud.IP-Veranstaltung *Grundlagen der Praktischen Informatik(Informatik II)* unter *Lernmodule* hinterlegt.

Sie können einen Test **nur einmal durchlaufen**. Sobald Sie einen Test starten steht Ihnen nur eine **begrenzte Zeit** zu Verfügung, um den Test zu bearbeiten.

Alle Punkte, die Sie beim Test erreichen, werden ihnen angerechnet.

Achtung

Manche Tests haben relativ lange Bearbeitungszeiten ($\geq$ 90 min.). Längere Inaktivitiät während dieser Zeit kann zum einem *timeout* und damit zu einem ungeplanten Ende des Tests führen.

Der Button *Test unterbrechen* verhindert den *timeout*, aber **stoppt nicht das weitere Ablaufen der Bearbeitungszeit**.

ILIAS 4-Minuten-Aufgaben – 12 Punkte

Absolvieren sie die Tests *GdPI 09 - 4 Minuten - Telematik - ...* für jeden dieser Tests haben Sie nur 4 Minuten Zeit.
(12 Punkte)

ILIAS – 25 Punkte

Vorbereitung

Bearbeiten Sie die Praktische Übung, Aufgabe 1 oder suchen Sie eine Webseite/installieren Sie ein Tool mit ähnlicher Funktionalität.

Telematik - Ethernet Paket

Absolvieren Sie den Test *GdPI 09 - Telematik - Ethernet Paket*.
(25 Punkte)

Übung

Abgabe bis Di., 27.06., 8 Uhr

Allgemein

Ihre Lösungen werden nur korrigiert, wenn Sie einer Übungsgruppe angehören, d.h. Teilnehmer einer Stud.IP-Veranstaltung *GdPI - Übung - ⟨⟨Termin⟩⟩* sind.

Nur wer in Ilias etwas abgibt kann Punkte bekommen. Siehe dazu die Anmerkungen auf Übungsblatt 02.

Benutzen Sie Markdown mit AsciiMath um die Lösung der Aufgaben zu strukturieren und zu kodieren. Fassen Sie die **Lösungen aller Aufgaben in einer Datei** zusammen.

Formatieren Sie Ihre Abgabe so, dass **die einzelnen Aufgaben klar durch passende Überschriften voneinander** getrennt sind.
Geben Sie die Datei über das Lernmodul *GdPI 09 - Markdown+AsciiMath* ab.

Aufgabe 1 – 8 Punkte

Transport

TCP und UDP sind beides Transportprotokolle. Für welche konkreten Dienste eignet sich

1. TCP besser als UDP
2. UDP besser als TCP

geben Sie jeweils zwei Beispiele an und keine kurze Begründung.
(8 Punkte)

Aufgabe 2 – 6 Punkte

Mars

Die Kameras der Rover, die sich auf dem Mars befinden, senden regelmäßig Bilder zur Erde (`https://www.dlr.de/next/desktopdefault.aspx/tabid-7379`).
Es handelt sich um eine Punkt-zu-Punkt-Übertragung. Eventuelle Wartezeiten, die durch Geräte oder das Protokoll entstehen, werden an anderer Stelle berechnet und deshalb hier nicht berücksichtigt.

Für diese Aufgabe wird Folgendes angenommen.

- Bildgröße 25 MiB (1 MiB = 2^{20} Byte)
- Datenrate = 196 kbps (kbps = 10^3 Bits pro Sekunde)
- Signalausbreitungsgeschwindigkeit = 299.792.458 m/s
- Abstand Erde und Mars = 81 Millionen Kilometer

Anmerkung

Der Abstand von Erde und Mars beziehen sich auf den 08.12.2022, an dem sich Sonne, Erde und Mars auf einer Linie befanden (Opposition `https://www.dlr.de/de/bilder/2020/04/oppositionen-von-erde-und-mars-1999-bis-2022`).

1. Um wieviele Sekunden verzögert trifft ein Signal von Mars auf der Erde ein (Ausbreitungsverzögerung)?
   (2 Punkte)

2. Wie lange dauert es das komplette Bild auf dem Mars abzuschicken (Übertragungsverzögerung)?
   (2 Punkte)

3. Wie lange dauert dauert die Übertragung eines kompletten Bildes von der Kamera auf dem Mars bis zum Kontrollzentrum auf der Erde (Latenz)?
   (2 Punkte)

Hinweis. Zwei Nachkommastellen Genauigkeit genügt.

Aufgabe 3 – 14 Punkte

Bitorientierte Protokolle

Bitorientierte Protokolle gehören überlicherweise zur Verbindungsschicht (*Data Link Layer*) und ermöglichen die Übertragung einer beliebige Anzahl von Bits.

Die Pakete des (fiktiven) *Simple Data Link Protocol* (*SDLP*) beginnen und enden mit derselben Blockbegrenzung (*opening/closing flag*) `01111110`. Auf das *opening flag* folgt eine 8-Bit Adresse (*address*), auf die wir hier nicht näher eingehen. Daran schließen sich n-Bit Daten (*data*) und ein 8-Bit Prüfwert der Daten (*frame check sequence*) an. Beendet wird das Paket mit dem *closing flag*.

8 bit	8 bit	n bit	8 bit	8 bit
`01111110`	address	data	frame check sequence	`01111110`

Die *frame check sequence* wird mit einer zyklischen Redundanzprüfung (*cyclic redundancy check*) mit 8 Bit (*CRC-8*) berechnet. Grundlage ist die Division mit Rest von binären Polynomen, das scheint aufwendig, ist aber algorithmisch relativ einfach umzusetzen.

Für den *CRC-8* des *SDLP* ist eine Bitfolge g = `111010101` festgelegt (das Generatorpolynom).

Initialisierung

1. Die Bitfolge der Daten ist das Zwischenergebnis c.

2. Ist die Länge von c kleiner 8, dann fülle c vorne mit Nullen auf 8 Bit auf.

3. Setze n auf die Länge (Anzahl Bits) von c.

4. Ergänze c am Ende um 8 Nullen.

Wiederhole n-mal

1. Beginnt c mit einer Eins verknüpfe die ersten 9 Bits von c und g bitweise mit Exclusive-Oder (XOR), die übrigen Bits von c bleiben unverändert.
2. Lösche die 0 am Anfang von c.

Ergebnis

Die Bitfolge c hat die Länge 8 und ist das Ergebnis

Beispiel

Ausgangssituation

g = `111010101`
c = `11000010` Länge $n = 8$

Initialisierung

c = `1100001000000000` 8 Nullen am Ende von c ergänzt

8 Wiederholungen

1. `1100001000000000` c
   `111010101` g
   `0010100010000000` neues c, Null vorne streichen

2. `0010100010000000` c beginnt mit Null
   `-`
   `010100010000000` neues c, Null vorne streichen

3. `10100010000000`
   `111010101`
   `001001000100000`

4. `1001000100000`
   `111010101`
   `000111101110000`

5. `111101110000`
   `111010101`
   `000111011000`

6. `00111011000`
   `-`
   `00111011000`

7. `0111011000`
   `-`
   `0111011000`

8. `111011000`
   `111010101`
   `00001101`

Ergebnis

`00001101` ist der Prüfwert nach *CRC-8* für die Daten `11000010`.

Aufgabe

An die Adresse `01100110` soll die Zeichenfolge `okay` gesendet werden, wobei die Zeichen jeweils mit **7-Bit** nach ASCII (*American Standard Code for Information Interchange*) codiert werden, siehe dazu z.B. `https://de.wikipedia.org/wiki/American_Standard_Code_for_Information_Interchange`.

1. Stellen Sie das *SDLP*-Paket als Bitfolge ohne *frame check sequence* dar.
   (4 Punkte)

2. Ergänzen Sie im *SDLP*-Paket aus Aufgabenteil 1 die *frame check sequence*.
   (5 Punkte)

3. Welche Probleme treten schon bei dem *SDLP*-Paket aus Aufgabenteil 1 auf und wie könnten eine Lösung dafür aussehen?
   (5 Punkte)

Praktische Übung

Abgabe der Prüfsumme bis Di., 27.06., 8 Uhr
Testat ab Mo., 04.07.

Hilfe zum Bearbeiten der praktischen Übungen können Sie grundsätzlich jeden Tag in den Rechnerübungen bekommen.

Am **Mo., 26.06.**, 10-12 Uhr online (3 Tutor*innen) und 18-20 Uhr in Präsenz (2 Tutor*innen), finden **keine Testate** statt. Diese Rechnerübungen sind ausschließlich **für Fragen** reserviert.

Abgabe der Prüfsumme

- Siehe vorherige Übungen.
- Übermitteln Sie die Prüfsumme mit dem Test *GdPI 09 - Testat*.

Aufgabe 1 – 10 Punkte

Vorbereitung

Informieren Sie sich über die *built-in functions* `int` und `bin` in der Python-Dokumentation (`https://docs.python.org/3/library/functions.html`).

Hex2Bin2Dec

Betrachten Sie folgendes Python-Skript, das alle auf der Kommandozeile übergebenen Argumente ausgibt.

```
1 #!/usr/bin/python3
2 import sys
3
4 def main():
5     for arg in sys.argv:
6         print(arg)
7
8 if __name__ == '__main__':
9      main()
```

Erweitern Sie die Funktion `main`, so dass dem Skript eine oder drei Argumente übergeben werden können, die wie folgt verarbeitet werden.

Das erste Argument wird als ganze Zahl x in Hexadezimal-Darstellung interpretiert.

- Wird nur eine Argument übergeben, wird x in Binär- und Dezimaldarstellung ausgegeben.
- Werden drei Argument übergeben, wird das zweite Argument als ganze Zahl a und das dritte als ganze Zahl b, jeweils in Dezimal-Darstellung, interpretiert.

  Die Bits von Index a bis $b-1$ der Binär-Darstellung von x sind die Binär-Darstellung der ganzen Zahl y.

  Die Binär- und Dezimaldarstellung von y wird ausgegeben.

Reagieren auf Fehler, z.B. falsche Anzahl von Argumenten, mit einer passenden Fehlermeldung und dem Programmende.
(10 Punkte)

Aufgabe 2 – 25 Punkte

Erfüllbarkeitstest

Stellen Sie Literale, Klauseln und Klauselmengen (Formeln in KNF) dar, wie auf Übungsblatt 08, Praktische Übung, Aufgabe 1.

1. Programmieren Sie ein Python-Skript, dass den *Algorithmus 2.33 (Erfüllbarkeitstest für aussagenlogische Formeln)* aus Kapitel 6 der Vorlesung implementiert.

2. Testen Sie Ihr Skript mit der Formel

   $$F = (X \vee \neg Y) \wedge (\neg X \vee \neg Y \vee \neg Z) \wedge (\neg X \vee Z) \wedge (X \vee Y \vee Z) \wedge (Y \vee \neg Z)$$

   aus Übung 07, Aufgabe 1.1.

3. Testen Sie Ihr Skript mit einer erfüllbaren Formel.

(25 Punkte)