Name of course

bei Prof. Dr. John Doe

Exercise Sheet 0

Ausarbeitung zum 01.01.1970 von:

Vorname Nachname	Vorname Nachname	Vorname Nachname	Vorname Nachname
1234567	1234567	1234567	1234567

an der **Universität Hamburg**

B.Sc. / Informatics / 12-345 Name of course

AUFGABE 0.1: PRIMZAHLÜBERPRÜFUNG.

(0.1.a) *Schreibe ein Python-Programm, das den Nutzer nach einer ganzen Zahl fragt und dann überprüft, ob die eingegebene Zahl eine Primzahl ist oder nicht. Eine Primzahl ist eine natürliche Zahl größer als 1, die nur durch 1 und sich selbst ohne Rest teilbar ist.*

```python
def ist_primzahl(zahl):
    if zahl ⩽ 1:
        return False
    elif zahl ⩽ 3:
        return True
    elif zahl % 2 == 0 or zahl % 3 == 0:
        return False
    i = 5
    while i * i ⩽ zahl:
        if zahl % i == 0 or zahl % (i + 2) == 0:
            return False
        i += 6
    return True

def main():
    try:
        eingabe = int(input("Geben Sie eine ganze Zahl ein: "))
        if ist_primzahl(eingabe):
            print(f"{eingabe} ist eine Primzahl.")
        else:
            print(f"{eingabe} ist keine Primzahl.")
    except ValueError:
        print("Ungültige Eingabe. Bitte geben Sie eine ganze Zahl ein.")

if __name__ == "__main__":
    main()
```

Dieses Programm definiert eine Funktion `ist_primzahl()`, die überprüft, ob eine Zahl eine Primzahl ist. In der `main`-Funktion wird der Benutzer nach einer Zahl gefragt und das Programm gibt dann aus, ob die Zahl eine Primzahl ist oder nicht.

(0.1.b) *Führen Sie eine Laufzeitanalyse für dieses Python-Programm durch. Beschreiben Sie die Komplexität des Algorithmus in Abhängigkeit von der Größe der übergebenen Zahl und erläutern Sie die beste, durchschnittliche und schlechteste Laufzeit des Programms.*

Die Laufzeitanalyse für den gegebenen Python-Code, der eine Zahl daraufhin überprüft, ob sie eine Primzahl ist oder nicht, kann durch die Betrachtung der Hauptfunktion `ist_primzahl()` erfolgen. Die Hauptlaufzeit des Programms hängt von der Größe der übergebenen Zahl ab.

Um die Laufzeit zu analysieren, betrachten wir die Anzahl der Iterationen in der Schleife innerhalb der Funktion `ist_primzahl()`. Diese Schleife läuft bis zur Quadratwurzel der übergebenen Zahl. Die Quadratwurzel ist ein guter Anhaltspunkt, da, wenn es einen Teiler gibt, dieser nicht größer als die Quadratwurzel der Zahl sein kann.

Die Komplexität der Funktion `ist_primzahl()` beträgt daher ungefähr $\mathcal{O}(\sqrt{N})$, wobei N die übergebene Zahl ist. Dies bedeutet, dass die Laufzeit des Programms proportional zur Quadratwurzel der übergebenen Zahl ist.

Hier ist eine Zusammenfassung der Laufzeitanalyse:

- Best Case Laufzeit: $\mathcal{O}(1)$ (wenn die Zahl 2 oder 3 ist, da die Funktion schnell feststellen kann, dass es sich um Primzahlen handelt)
- Average Case Laufzeit: $\mathcal{O}(\sqrt{N})$
- Worst Case Laufzeit: $\mathcal{O}(\sqrt{N})$ (wenn die übergebene Zahl eine sehr große Primzahl ist)

AUFGABE 0.2: MATHEMATIK.

Geben Sie einen coolen mathematischen Ausdruck an.

$$\int_{-\infty}^{\infty} \mathrm{e}^{-x^2} \cdot \left(1 + \frac{\sin^2(x)}{x^2}\right) \mathrm{d}x = \frac{\pi}{\sqrt{2}} \cdot \sum_{n=0}^{\infty} (-1)^n \cdot \frac{(2n)!}{2^{2n} \cdot (n!)^2}.$$