1 C-Programm

Schreiben Sie ein Programm *ringbuffer* in C, das einen Ringbuffer mit $N = 20$ Elementen vom Typ `char` abbildet. Das Programm soll einen String als Benutzereingabe über die Konsole einlesen und den Buffer füllen. Sobald 10 neue Zeichen eingegeben wurden, werden die ersten 5 Zeichen im Buffer auf der Konsole ausgegeben.

Beispiel für eine Verwendung:

1. Der Ringbuffer ist leer und fasst 20 Elemente.
2. Die Benutzereingabe lautet: `Dies␣ist␣eine␣Beispieleingabe`
3. Der Inhalt des Ringbuffers ist dann nach den ersten 10 Zeichen: `Dies␣ist␣e`
4. Es werden fünf Zeichen ausgegeben: `Dies␣`
5. Der Inhalt des Ringbuffers ist nun: `ist␣e`
6. Es werden weitere 10 Zeichen eingelesen: `ist␣eine Beispi`
7. Wieder werden fünf Zeichen ausgegeben: `ist␣e`
8. usw.

Beachten Sie, dass die Anzahl der belegten Elemente im Ringbuffer wächst und die Kapazität übersteigt (Anzahl Eingabezeichen > Anzahl Ausgabezeichen); behandeln Sie diesen Überlauf durch das Überschreiben des "ältesten" Inhalts.

2 C-Deklarationen

Erklären Sie folgende C-Deklarationen:

Beispiel: `const int* (*x)()` – x ist ein Zeiger auf eine Funktion die keine Argumente nimmt und einen Zeiger auf einen const int zurückgibt.

1. `void* f[2];`
2. `int* const (*g)();`
3. `const char* const (*(*h[3][2])(const char*));`
4. `struct a i[2];`
5. `struct a(*j)(struct a* const);`

Überprüfen Sie das Ergebnis auf `http://cdecl.org/`.

3 Rekursion

Gegeben ist eine iterative Implementierung zur Berechnung von $f(x, n) = x^n$ und ein Beispiel für den Aufruf. Schreiben Sie eine endrekursive Variante zur Berechnung dieser Funktion.

```
int f(int x, int n) {
        int y = x;
        for (int i=1; i<n; i++) y = y * x;
        return y;
}

f(3,4);
```

4 Zweierkomplement

Schreiben Sie das Ergebnis der folgenden Rechnungen als Dezimalzahl mit Vorzeichen (alle Zahlen in 8 Bit Zweierkomplementdarstellung):

1. 0xA0 + 0x10
2. 0xA0 - 0x10
3. 0xFF + 0xAA
4. 0x7F + 0x7F

5 Sortieren

Gegeben ist folgendes Array:

```
int* a[100];
```

Das Array kann auch `NULL`-Werte beinhalten. Diese sollen nach dem Sortieren am Ende liegen.

Schreiben Sie eine Vergleichsfunktion für `qsort` aus der `stdlib`, um dieses Array *absteigend* zu sortieren.

6 Ausführung in konstanter Zeit

Gegeben ist folgende Funktion zur Berechnung der Summe aller Werte größer oder gleich 10 in einem Array. Werte kleiner 10 werden in der Berechnung der Summe nicht berücksichtigt. Schreiben Sie eine Variante dieser Funktion die unabhängig von Eingabe und Ausgabe in konstanter Zeit terminiert.

```
unsigned int array_sum(unsigned int* data, int len) {
        unsigned int sum = 0;
        for(int i=0; i<len; i++)
                if (data[i] >= 10)
                        sum += data[i];
        return sum;
}
```

7 Scheduling

Es sind vier Prozesse (PID) mit Ankunftszeit und Ausführungsdauer gegeben.

PID	Ankunft	Dauer
1	0	10
2	10	15
3	10	30
4	20	10

Der Prozess mit der PID 3 wartet nach der Ausführung von 5 Zeiteinheiten auf einen Hardwareinterrupt der nach weiteren 5 Zeiteinheiten auftritt.

Skizzieren Sie den zeitlichen Ablauf für Scheduling mit *einer* CPU mit den Strategien (a) Shortest Job First; (b) Shortest Remaining Time; und (c) Round Robin (mit Zeitquantum 10 Zeiteinheiten).

Hinweis: Skizzieren Sie zunächst den zeitlichen Ablauf wenn jeder Prozess auf einer eigenen CPU ausgeführt wird. Skizzieren Sie danach den zeitlichen Ablauf für verschiedene Scheduling-Strategien wenn nur eine CPU zur Verfügung steht.

8 Theoriefragen

1. Erklären Sie den Unterschied zwischen Prozessen und Threads. Welche Elemente sind für jeden Prozess vorhanden und welche für jeden Thread?
2. Geben Sie ein einfaches Beispiel für die Verwendung von `fork` aus der POSIX-Programmierschnittstelle.
3. Erklären Sie den Unterschied zwischen `union` und `struct` in C.
4. Was versteht man unter einem Deadlock? Welche Bedingungen sind für einen Auftritt erforderlich.
5. Geben Sie die Komplexität für den Quick-Sort-Algorithmus und für den Bubble-Sort-Algorithmus (jeweils average case) in Groß-O-Notation und in Abhängigkeit der Anzahl n der zu sortierenden Elemente an.