hyperstone Programmier-Challenge

Um mit dieser Challenge am Gewinnspiel teil zunehmen muss diese bis zum 16.Dezember 2024 abgegeben werden. Entweder digital an jzinn@hyperstone.com.

E-Mail Adresse

Aufgabe F1:

Gegeben ist die Funktion `getMostSignificantBit(unsigned int mask)`, welche den Index des höchsten gesetzten Bits einer Maske zurückgibt:

```
static unsigned int getMostSignificantBit(unsigned int mask)
{
  return ((sizeof(unsigned int) * 8) - 1) - __builtin_clz(mask);
}

Int main(int argc, char** argv)
{
   unsigned int bitMaske = (1<<23) | (1<<10) | (1<<13);
   // insert code here.
}
```

Erweitere die Funktion `main` so, dass alle gesetzten Bits einer Maske ausgegeben werden. Für die Maske oben sollte die Ausgabe wie folgt sein:

```
Index 23 set

Index 13 set

Index 10 set
```

Aufgabe F2:

Die folgende Aufzählung ist gegeben:

```
typedef enum
{
  verySpecial,
  verySignificant,
  moderatelySignificant,
  slightlySignificant,
  slightlyUnspecial,
} SpecialValues;
```

a.) Was macht folgende Funktion?

```
static inline unsigned int fancyFunction(SpecialValues val)
{
  return (((1 << verySignificant) |
           (1 << moderatelySignificant) |
           (1 << slightlySignificant)) >> val) & 1;
}
```

b.) Warum funktioniert folgende Funktion nicht wie erwartet? Was muss geändert werden, damit die Funktion korrekt ist?

```
static inline unsigned int anotherFancyFunction(SpecialValues val)
{
  return (val & (verySignificant | moderatelySignificant | slightlySignificant));
}
```

Aufgabe F3:

Schreibe eine Funktion, die zufällig Werte mit folgenden Wahrscheinlichkeiten zurückgibt:

32%	42
22%	13
11%	37
35%	Einen Integer im Intervall [0,10) (normalverteilt)

Nimm an, dass die vordefinierte Funktion `unsigned int getRandomBoundedInteger(unsigned int maximum)` normalverteilte Integer zwischen 0 und Maximum (exklusive dem Maximum) zurückgibt.