AUFGABE 1

Was gibt folgendes Programm am Bildschirm aus?

```
# include <stdio.h>

# define PR(format, value) printf(#value" = %"#format"\t", (value))
# define NL putchar('\n');
# define PRINT1(f,x1) PR(f,x1), NL
# define PRINT2(f,x1,x2) PR(f,x1), PRINT1(f,x2)

int main (void)
{
	static struct S1
	{
		char c[4], *s;
	}
	s1 = {"789", "XYZ"};

	static struct S2
	{
		char *cp;
		struct S1 *ss1;
	}
	s2 = {"abc", &s1}, *s2_ptr = &s2;

	PRINT2(c, *s2.cp, *s2.ss1->s);
	PRINT2(s, ++s2.cp, ++s2.ss1->s);
	PRINT2(s, s2_ptr->ss1->c, ++s2_ptr->ss1->s);
	PRINT2(c, *(*s2.ss1).c, *(*s2.ss1).s);

	return 0;
}
```

AUFGABE 2

Was gibt folgendes Programm am Bildschirm aus?
Der Aufruf erfolgt mit dem Kommandozeilenparameter `GREBNRUEN`

```
# include <stdio.h>
# include <string.h>

int main (int argc, char *argv[])
{
	char **s, *t;

	if (argc > 1)
	{
		s = argv+1;
		t = *s + strlen(*s);
		while (*s - --t)
		{
			printf("%c", *t);
		}
		printf("%c\n", *t);
	}
	return 0;
}
```

AUFGABE 3

Schreiben Sie eine Funktion str2upper, welche sämtliche Kleinbuchstaben in einem ihr übergebenen String in Großbuchstaben umwandelt. Umlaute sollen nicht berücksichtigt werden.

AUFGABE 4:

Implementieren Sie eine Funktion GetLastElement, welche das letzte Element einer verketteten Liste zurückliefert.

AUFGABE 5:

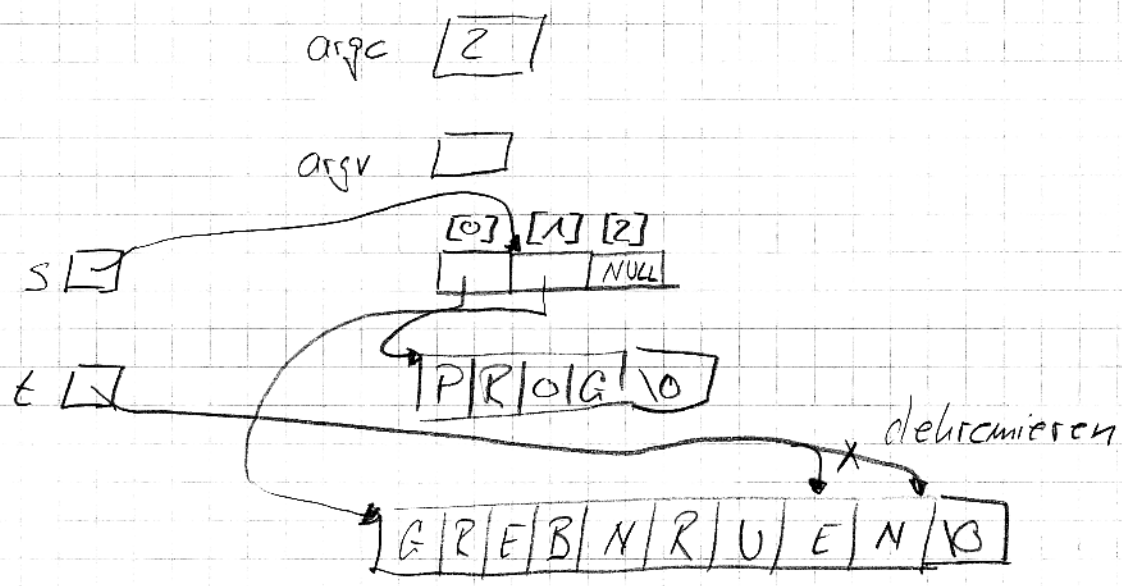
Schreiben Sie die Funktion KnotenAnzahl. Die Funktion soll die Anzahl der Knoten in einem binären Baum ermitteln und zurückliefern.

AUFGABE 6:

Schreiben Sie ein Programm, daß die letzten x-Zeichen aus einer Datei ausliest und auf den Bildschirm schreibt. Als Kommandozeilenparameter sind dabei die Anzahl der Zeichen (x) und der Dateiname der Datei, aus der kopiert werden soll, notwendig.

Aufgabe 2

PROG GREBNRUEN



⇒ NUERNBERG

Aufgabe 3

```
int str2upper (char *s)
{
   int i = 0;
   while (*s)
   {
      *s = (((*s >= 'a') && (*s <= 'z')) ? ++i, (*s - 'a')
                                          + 'A' : *s);
      s++;
   }
   return i;
}
```

oder:

```
if ((*s >= 'a') && (*s <= 'z'))
{
   *s = (*s - 'a') + 'A';
   i++;
}
s++
```

Aufgabe 4:

```
LISTENELEMENT *GetLastElement ( LISTENELEMENT **Liste )
{
    if ( *Liste == NULL)
        return 0;

    else if ( (**Liste).Naechstes == NULL )
    {
        LISTENELEMENT *Element;
        Element =*Liste;
        *Liste = NULL;
        return Element;
    }
    else
        return GetLastElement (&(**Liste).Naechstes);

}
```

Aufgabe 5:

```
unsigned int KnotenAnzahl ( BAUMKNOTENELEMENT *Wurzel)
{
    if ( Wurzel == NULL)
        return 0;
    else
        return  KnotenAnzahl ((*Wurzel).Linker) +
                KnotenAnzahl ((*Wurzel).Rechter) +
                1;
}
```

Aufgabe 6:

```
int main (int argc, char **argv)
{ FILE *fp; int zahl;
  if (argc < 3)                    // argc>3
  {
    fprintf (stderr, "Syntax: TYPEEND Dateiname Zeichen-
                                          anzahl \n");
    exit (1);
  }

  if ((fp = fopen (argv[1], "r")) == NULL
  {
    fprintf (stderr, "Datei %s lässt sich nicht öffnen \n", argv
                                                      [1]);
    exit (2);
  }

  if ((sscanf (argv[2], "%d", &zahl)) != 1)
  {
    fprintf (stderr, "Es wurde keine Zahl übergeben \n");
    exit (3);
  }

  if (zahl <= 0)
  {
    fprintf (stderr, "zahl <= 0 \n");
    exit (4);
  }

  fseek (fp, zahl, SEEK_END);
  filecopy (fp, stdout);
  return 0;                        fclose (fp);
}
```

siehe KR Fkt. filecopy

```
void filecopy (fp, stdout)
{ int c;
  while ((c = getc
          (fp)) != EOF)
    putc (c, stdout);
}
```