Übung 2.1 - String Reverse

```
/**============================================================================
Hochschule Luzern - Technik & Architektur            Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                              http://hslu.ximit.ch
------------------------------------------------------------------------------
Name        : Uebung_2-1.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-09-23v1.0
Description : Reverse
              Schreiben Sie eine Funktion reverse(s),
              die die Zeichenkette s zeilenweise umkehrt.
============================================================================*/

#include <stdio.h>
#include <stdlib.h>



/**
* Liest eine Zeile von maximal limit Zeichen ein.
* Die Zeichen werden (inklusive Zeilenende-Zeichen) im übergebenen * Vektor s /0 terminiert abgelegt.
* @param s Zeiger auf den Vektor zum Speichern der Eingabe
* @param limit Maximale Grösse des Vektors
* @return Anzahl eingelesene Zeichen
*/
int readLine(char s[], int limit);

/**
* Kehrt den Input Stream um speichert ihn in out.
* @param in String Input
* @param out Input String Reverse
*/
void reverseString(char in[], char out[], int len);


/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{

        char stringInput[100], stringReverse[100];
        int readCount;

        // Read Input String
        readCount = readLine(stringInput, 100);
        printf("Anz. Zeichen: %d\nInput:  %s", readCount, stringInput);
        //Reverse String
        reverseString(stringInput, stringReverse, readCount-1);
        //Output
        printf("Output: %s\n", stringReverse);







        fflush(stdin); //flushes the input stream and gets rid of '\n'
        printf("\n\nPress [Enter] to exit...");
        getchar();
        return(EXIT_SUCCESS);
}


int readLine(char s[], int limit)
{
        int i = 0;
        int c;
        c = getchar();                  /* Buchstabe einlesen */
        while((c != EOF) &&             /* Ende File ... */
          (c != '\n') &&/* oder Ende Zeile ... */
                   (i < limit - 1))     /* oder Limite des Speichers? */
        {
                s[i] = c;
                i++;
```

```
                c = getchar();
        }
        if (c == '\n')
        {
                s[i] = '\n';            /* Zeilenumbruch anfügen! */
                i++;
        }
        s[i] = '\0';                    /* Zeichenkette-Ende anfügen! */
        return i;                               /* Anzahl gelesene Zeichen zurückgeben */
}

void reverseString(char in[], char out[], int len)
{
        int i = 0;
        for (; i<len; i++)
        {
                out[i] = in[len-i-1];
        }
        out[i] = '\0';
}
```

Übung 2.2 - Integer to String

```
/**============================================================================
Hochschule Luzern - Technik & Architektur            Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                              http://hslu.ximit.ch
------------------------------------------------------------------------------
Name        : Uebung_2-2.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-09-23v1.0
Description : Integer to String
              Schreiben Sie eine Funktion char* itoa(int i), welche die
              übergebene Integerzahl in dezimaler Darstellung als
              Zeichenkette zurück gibt.
============================================================================*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

/**
 * Berechnet die Anzahl stellen einer Zahl
 * @param number Zahl von der die anzahl Stellen gesucht sind
 * @return Anzahl Stellen
 */
int getDigitCount(int number);

/**
 * Gibt die Zahl als String zurück
 * @param i Zahl als Integer
 * @return Zahl als String
 */
char* itoa(int i);

/**
 * Kehrt den Input Stream um speichert ihn in out.
 * @param in String Input
 * @return String Reverse
 */
char* reverseString(char in[]);


/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{
        int wert;
        char *result;

        // Wert einlesen
        printf("Wert als Integer (auch negativ): ");
        scanf("%d", &wert);

        // Integer to ASCII
        result = itoa(wert);

        printf("Result: %s\n", result);

        fflush(stdin); //flushes the input stream and gets rid of '\n'
        printf("\n\nPress [Enter] to exit...");
        getchar();
        return(EXIT_SUCCESS);
}

int getDigitCount(int number) {
    if(number != 0) {
        return (int)log10(abs(number)) + 1;
    } else {
        return 1;
    }
}

char* itoa(int i)
{
        int countDigit = 0;
        int pos = 0;
        int isPositive = 1;
```

```
        char *res;

        // Anzahl Stellen ermitteln
        countDigit = getDigitCount(i);
        // printf("GetDigitCount(%d): %d\n", i, countDigit);
        if (i < 0)
        {
                isPositive = 0;
                countDigit++;
                i = abs(i);
        }

        // Speicher für String reservieren
        res = malloc(sizeof(char) * countDigit + 1);
        if (res)
        {
                while(i != 0)
                {
                        res[pos] = (char) (i % 10 + 48); // + 48 für ASCII Wert
                        i /= 10;
                        pos++;
                }
                if (!isPositive)
                {
                        res[pos] = '-';
                        pos++;
                }
                res[pos] = '\0';
                res = reverseString(res);
        }
        else
        {
                printf("Error bei Speicherallokation...\n");
        }

        return res;
}

char* reverseString(char in[])
{
        int len = strlen(in);
        int i = 0;
        char tmp;
        for (; i<(len/2); i++)
        {
                tmp = in[i];
                in[i] = in[len-i-1];
                in[len-i-1] = tmp;
        }
        return in;
}
```

Übung 2.3 - Vektoren, malloc, realloc

```
/**============================================================================
Hochschule Luzern - Technik & Architektur            Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer, Mike Estermann
------------------------------------------------------------------------------
Name        : Uebung_2-3.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
              Mike Estermann <mike.estermann@stud.hslu.ch>
Version     : 2012-09-23v1.0
Description : Entwurf Übung mit Vektoren und malloc / realloc
============================================================================*/

#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include <time.h>

/**
 * Memory Allozieren
 * @param size Anzahl MB die alliziert werden sollen
 * @return int-Vector
 */
int* allocate(long size);

/**
 * Sensless-Methode um den Vector zu benutzen.
 * Random Nummern speichern und wieder lesen.
 * @param vector Pointer auf den Vector
 */
void useVector(int *vector);


/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{
        // Variablen definieren
        int wert;
        long size;
        int *vector;

        // get size
        printf("Wie viel MB soll alloziert werden: ");
        scanf("%d", &wert);
        size = wert * 1024 * 1024; // Eingaben in MB umrechnen
        fflush(stdin); //flushes the input stream and gets rid of '\n'

        // allocate vector
        vector = allocate(size);

        // use vector
        printf("\n[Enter] druecken um den Speicher zu benutzen...");
        getchar();
        useVector(vector);

        // free memory
        printf("\n[Enter] druecken um den Speicher freizugeben...");
        getchar();
        free(vector);

        // exit
        printf("\n\nPress [Enter] to exit...");
        getchar();
        return(EXIT_SUCCESS);
}


/**
 * Memory Allozieren
 * @param size Anzahl MB die alliziert werden sollen
 * @return int-Vector
 */
int* allocate(long size)
{
        // Variablen definieren
        const int steps = 20;
```

```
	int index = 0;
	int *vector;

	// calc size based on integer-size
	size = size / sizeof(int);

	// Allocate the memory
	for (; index < steps; index++)
	{
		// Allocate or reallocate memory
		vector = (int *)((index == 0) ? malloc(sizeof(int) * size / steps) : realloc(vector,
(sizeof(int) * size / steps) * index));

		if (vector == NULL)
		{
			// Error
			perror("\nNicht genug Speicher vorhanden!\n\n");
			// Exit
			exit(EXIT_FAILURE);
		}

		// delay for demo
		Sleep(500);
		printf("Memory allocated: %d\n", ((sizeof(int) * size * (index + 1)) / steps));
	}

	// return pointer of the new vector
	return vector;
}

/**
 * Sensless-Methode um den Vector zu benutzen.
 * Random Nummern speichern und wieder lesen.
 * @param vector Pointer auf den Vector
 */
void useVector(int *vector)
{
	// Variablen definieren
	int index;
	int countNumbers = 10;

	// Initialize random generator
	srand(time(NULL));

	// generate random numbers
	for (index = 0; index < countNumbers; index++)
	{
		*(vector + index) = rand();
	}

	// print numbers
	for(index = 0; index < countNumbers; index++)
	{
		printf("%i. %i\n", index + 1, *(vector + index));
	}
}
```

Übung 2.4 - Enum to String

```
/**============================================================================
Hochschule Luzern - Technik & Architektur            Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                              http://hslu.ximit.ch
------------------------------------------------------------------------------
Name        : Uebung_2-4.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-09-23v1.0
Description : Enum to String
============================================================================*/

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

typedef enum Color_
{
        BLACK,  // Wert 0
        RED,    // Wert 1
        ORANGE, // Wert 2, usw
        YELLOW,
        GREEN,
        BLUE,
        VIOLET
} Color_t;

/**
 * Gibt den Namen einer Farbe zurück
 * @param color ENUM der Farbe
 * @return String mit dem Namen der Farbe
 */
char* getColorName(Color_t color);

/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{
        // use return value
        char *colorName;
        colorName = getColorName(RED);
        printf("Color: %s\n", colorName);

        // pass direct to printf()
        printf("Color: %s\n", getColorName(YELLOW));

        // use Number insted of ENUM
        printf("Color: %s\n", getColorName(0));

        // unknow Color
        printf("Color: %s\n", getColorName(99));



        fflush(stdin); //flushes the input stream and gets rid of '\n'
        printf("\n\nPress [Enter] to exit...");
        getchar();
        return(EXIT_SUCCESS);
}


/**
 * Gibt den Namen einer Farbe zurück
 * @param color ENUM der Farbe
 * @return String mit dem Namen der Farbe
 */
char* getColorName(Color_t color)
{
        const int COLORNAME_MAX_LEN = 10; // Max. 9 Zeichen für den Namen + \0 !
        char* colorName = malloc(sizeof(char) * COLORNAME_MAX_LEN);
        if (colorName) // Memory allocation successfully
        {
                switch (color) {
                case BLACK:
                        strcpy(colorName, "Black");
                        break;
                case RED:
```

```
                        strcpy(colorName, "Red");
                        break;
                case ORANGE:
                        strcpy(colorName, "Orange");
                        break;
                case YELLOW:
                        strcpy(colorName, "Yellow");
                        break;
                case GREEN:
                        strcpy(colorName, "Green");
                        break;
                case BLUE:
                        strcpy(colorName, "Blue");
                        break;
                case VIOLET:
                        strcpy(colorName, "Violet");
                        break;
                default:
                        strcpy(colorName, "UNKNOW");
                        break;
                }
        }
        else
        {
                // Memory allocation failed!
                perror("Error allocate Memory!");
                exit(EXIT_FAILURE);
        }
        return colorName;
}
```

Übung 2.5 - strcat

```
/**============================================================================================
Hochschule Luzern - Technik & Architektur                                  Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                   http://hslu.ximit.ch
--------------------------------------------------------------------------------------------------
Name        : Uebung_2-5.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-09-23v1.0
Description : strcat
              Schreiben Sie die Funktion mystrcat(s, t) (Sting concatenate) mit Zeiger, welche
              eine Zeichenkette t an das Ende der Zeichenkette s kopiert.
              Zur Erinnerung: Das '/0' Zeichen markiert das Ende einer Zeichenkette.
============================================================================================*/

#include <stdio.h>
#include <stdlib.h>

/**
 * Kopiert String t ans Ende von String s.
 * Vorsicht: String s muss genügend gross sein!
 */
void mystrcat(char *s, const char *t);

/**
 * The main procedure to run the application.
 * @param argc Number of command line arguments
 * @param argv The forwarded command line arguments
 * @return Application return (error) code
 */
int main(int argc, char** argv)
{
        char input1[256];
        char input2[128];

        printf("Erste Zeichenkette: ");
        scanf("%s", input1);

        printf("\nZweite Zeichenkette: ");
        scanf("%s", input2);

        mystrcat(input1, input2);
        printf("\nErgebnis von strcat: %s\n", input1);


        fflush(stdin); //flushes the input stream and gets rid of '\n'
        printf("\n\nPress [Enter] to exit...");
        getchar();
        return(EXIT_SUCCESS);
}

/**
 * Kopiert String t ans Ende von String s.
 * Vorsicht: String s muss genügend gross sein!
 */
void mystrcat(char *s, const char *t)
{
        while (*s) //solange kein \0 gefunden
                s++;   // bis ans ende von s gehen
        while (*s++ = *t++) // t an s anhängen
                ; // work done in der while-bedingung
}
```

Übung 2.6 - Fibonacci Zahlen

```
/**============================================================================================
Hochschule Luzern - Technik & Architektur                                  Mikrocontroller - HS2012
Copyright 2012 Felix Rohrer                                                   http://hslu.ximit.ch
--------------------------------------------------------------------------------------------
Name        : Uebung_2-6.c
Author      : Felix Rohrer <felix.rohrer@stud.hslu.ch>
Version     : 2012-09-23v1.0
Description : Fibonacci Zahlen
============================================================================================*/


#include <stdio.h>
#include <stdlib.h>

/**
 * Berechnet die Fibonacci Zahlen.
 * Musterlösung von HSLU, Peter Sollberger
 * @author Peter Sollberger
 */
int main(int argc, char** argv)
{
	int number, i, count;
	int* fibonaccis;

	// Wert einlesen
	printf("Wieviele Fibonacci-Zahlen wollen Sie berechnen? ");
	scanf("%d", &number);
	printf("\n");

	count = number + 1;
	fibonaccis = malloc(sizeof (int) * count);
	if (fibonaccis)
	{
		// Calc Fibonacci-Zahlen
		fibonaccis[0] = 0;
		fibonaccis[1] = 1;
		for (i = 2; i < count; i++)
		{
			fibonaccis[i] = fibonaccis[i-1] + fibonaccis[i-2];
		}

		// Print
		printf("Die Fibonacci Zahlen von 0 bis %d lauten:\n", number);
		for (i = 0; i < count; i++)
		{
			printf("%d\n", fibonaccis[i]);
		}

		// Free Memory
		free(fibonaccis);
	}
	else
	{
		printf("Fehler in Speicherallokation\n");
	}

	fflush(stdin); //flushes the input stream and gets rid of '\n'
	printf("\n\nPress [Enter] to exit...");
	getchar();
	return(EXIT_SUCCESS);
}
```