0x0C-more\_alloc\_free

===== 0-malloc\_checked.c ======

#include "main.h"

/\*\*

\* malloc\_checked - cause normal process termination with a status value of 98

\*

\* @b: allocated memory

\*

\* Return: a pointer to the allocated memory

\*/

void \*malloc\_checked(unsigned int b)

{

   void \*i;

   i = malloc(b);

   if (i == NULL)

       exit(98);

   return (i);

}

======= 1-string\_nconcat.c =======

#include "main.h"

/\*\*

\* string\_nconcat - a function that concatenates two strings.

\*

\* @s1: first char

\* @s2: secound char

\* @n: unsigned int

\*

\* Return: If the function fails, it should return NULL

\*/

char \*string\_nconcat(char \*s1, char \*s2, unsigned int n)

{

   unsigned int x, y, z;

   char \*s;

   if (s1 == NULL)

   {

       x = 0;

   }

   else

   {

       for (x = 0; s1[x]; ++x)

           ;

   }

   if (s2 == NULL)

   {

       y = 0;

   }

   else

   {

       for (y = 0; s2[y]; ++y)

           ;

   }

   if (y > n)

       y = n;

   s = malloc(sizeof(char) \* (x + y + 1));

   if (s == NULL)

       return (NULL);

   for (z = 0; z < x; z++)

       s[z] = s1[z];

   for (z = 0; z < y; z++)

       s[z + x] = s2[z];

   s[x + y] = '\0';

   return (s);

}

== == == == = 2-calloc.c ====

#include "main.h"

/\*\*

\* \_calloc - allocates memory for an array of @nmemb elements of

\* @size bytes each and returns a pointer to the allocated memory.

\*

\* @nmemb: allocate memory for array

\* @size: allocate element of size bytes

\*

\* Return: pointer to the allocated memory.

\*/

void \* \_calloc(unsigned int nmemb, unsigned int size)

{

   char \*a;

   unsigned int b;

   if (nmemb == 0 || size == 0)

       return (NULL);

   a = malloc(nmemb \* size);

   if (a == NULL)

       return (NULL);

   for (b = 0; b < (nmemb \* size); b++)

       a[b] = 0;

   return (a);

}

======= 3-array\_range.c ======

#include <stdlib.h>

#include "main.h"

/\*\*

\* \*array\_range - creates an array of integers

\* @min: minimum range of values stored

\* @max: maximum range of values stored and number of elements

\*

\* Return: pointer to the new array

\*/

int \*array\_range(int min, int max)

{

   int \*ptr;

   int i, size;

   if (min > max)

       return (NULL);

   size = max - min + 1;

   ptr = malloc(sizeof(int) \* size);

   if (ptr == NULL)

       return (NULL);

   for (i = 0; min <= max; i++)

       ptr[i] = min++;

   return (ptr);

}

== == == == == 100 - realloc.c == == == ==

#include "main.h"

/\*\*

\* \_realloc - reallocates a memory block using malloc and free

\*

\* @ptr: pointer to the memory previously allocated

\* @old\_size:is the size, in bytes, of the allocated space for ptr

\* @new\_size: the new size, in bytes of the new memory block

\*

\* Return: pointer allocate new size memory, or NULL

\*/

void \*\_realloc(void \*ptr, unsigned int old\_size, unsigned int new\_size)

{

   char \*p;

   unsigned int i, n = new\_size;

   char \*oldp = ptr;

   if (ptr == NULL)

   {

       p = malloc(new\_size);

       return (p);

   }

   else if (new\_size == 0)

   {

       free(ptr);

       return (NULL);

   }

   else if (new\_size == old\_size)

       return (ptr);

   p = malloc(new\_size);

   if (p == NULL)

       return (NULL);

   if (new\_size > old\_size)

       n = old\_size;

   for (i = 0; i < n; i++)

       p[i] = oldp[i];

   free(ptr);

   return (p);

}

== == == == 101 - mul.c == ==

#include "main.h"

   /\*\*

    \* \_print - moves a string one place to the left and prints the string

    \* @str: string to move

    \* @l: size of string

    \*

    \* Return: void

    \*/

   void \_print(char \*str, int l)

{

   int i, j;

   i = j = 0;

   while (i < l)

   {

       if (str[i] != '0')

           j = 1;

       if (j || i == l - 1)

           \_putchar(str[i]);

       i++;

   }

   \_putchar('\n');

   free(str);

}

/\*\*

\* mul - multiplies a char with a string and places the answer into dest

\* @n: char to multiply

\* @num: string to multiply

\* @num\_index: last non NULL index of num

\* @dest: destination of multiplication

\* @dest\_index: highest index to start addition

\*

\* Return: pointer to dest, or NULL on failure

\*/

char \*mul(char n, char \*num, int num\_index, char \*dest, int dest\_index)

{

   int j, k, mul, mulrem, add, addrem;

   mulrem = addrem = 0;

   for (j = num\_index, k = dest\_index; j >= 0; j--, k--)

   {

       mul = (n - '0') \* (num[j] - '0') + mulrem;

       mulrem = mul / 10;

       add = (dest[k] - '0') + (mul % 10) + addrem;

       addrem = add / 10;

       dest[k] = add % 10 + '0';

   }

   for (addrem += mulrem; k >= 0 && addrem; k--)

   {

       add = (dest[k] - '0') + addrem;

       addrem = add / 10;

       dest[k] = add % 10 + '0';

   }

   if (addrem)

   {

       return (NULL);

   }

   return (dest);

}

/\*\*

\* check\_for\_digits - checks the arguments to ensure they are digits

\* @av: pointer to arguments

\*

\* Return: 0 if digits, 1 if not

\*/

int check\_for\_digits(char \*\*av)

{

   int i, j;

   for (i = 1; i < 3; i++)

   {

       for (j = 0; av[i][j]; j++)

       {

           if (av[i][j] < '0' || av[i][j] > '9')

               return (1);

       }

   }

   return (0);

}

/\*\*

\* init - initializes a string

\* @str: sting to initialize

\* @l: length of strinf

\*

\* Return: void

\*/

void init(char \*str, int l)

{

   int i;

   for (i = 0; i < l; i++)

       str[i] = '0';

   str[i] = '\0';

}

/\*\*

\* main - multiply two numbers

\* @argc: number of arguments

\* @argv: argument vector

\*

\* Return: zero, or exit status of 98 if failure

\*/

int main(int argc, char \*argv[])

{

   int l1, l2, ln, ti, i;

   char \*a;

   char \*t;

   char e[] = "Error\n";

   if (argc != 3 || check\_for\_digits(argv))

   {

       for (ti = 0; e[ti]; ti++)

           \_putchar(e[ti]);

       exit(98);

   }

   for (l1 = 0; argv[1][l1]; l1++)

       ;

   for (l2 = 0; argv[2][l2]; l2++)

       ;

   ln = l1 + l2 + 1;

   a = malloc(ln \* sizeof(char));

   if (a == NULL)

   {

       for (ti = 0; e[ti]; ti++)

           \_putchar(e[ti]);

       exit(98);

   }

   init(a, ln - 1);

   for (ti = l2 - 1, i = 0; ti >= 0; ti--, i++)

   {

       t = mul(argv[2][ti], argv[1], l1 - 1, a, (ln - 2) - i);

       if (t == NULL)

       {

           for (ti = 0; e[ti]; ti++)

               \_putchar(e[ti]);

           free(a);

           exit(98);

       }

   }

   \_print(a, ln - 1);

   return (0);

}

== == == == == = main.h == == == =

#ifndef MAIN\_H

#define MAIN\_H

void \_print(char \*str, int l);

char \*mul(char n, char \*num, int num\_index, char \*dest, int dest\_index);

int check\_for\_digits(char \*\*av);

void init(char \*str, int l);

#include <stdio.h>

#include <stdlib.h>

int \_putchar(char c);

void \*malloc\_checked(unsigned int b);

char \*string\_nconcat(char \*s1, char \*s2, unsigned int n);

void \*\_calloc(unsigned int nmemb, unsigned int size);

int \*array\_range(int min, int max);

void \*\_realloc(void \*ptr, unsigned int old\_size, unsigned int new\_size);

#endif