my\_string.h

#ifndef MY\_STRING\_H

#define MY\_STRING\_H

typedef void\* MY\_STRING;

MY\_STRING my\_string\_init\_default(void);

MY\_STRING my\_string\_init\_c\_string(const char\* c\_string);

int my\_string\_get\_capacity(MY\_STRING hMy\_string);

int my\_string\_get\_size(MY\_STRING hMy\_string);

int my\_string\_compare(MY\_STRING hLeft\_string, MY\_STRING hRight\_string);

void my\_string\_destroy(MY\_STRING\* phMy\_string);

#endif

my\_string.c

#include <stdlib.h>

#include <stdio.h>

#include "my\_string.h"

typedef struct my\_string

{

int size;

int capacity;

char\* data;

}My\_string;

MY\_STRING my\_string\_init\_default(void){

My\_string\* pString=(My\_string\*) malloc(sizeof(My\_string));

pString -> size = 0;

pString -> capacity = 7;

pString -> data = (char\*) malloc(sizeof(char)\*pString -> capacity);

if(pString -> data == NULL)

{ free(pString);

pString = NULL;

}

return (MY\_STRING)pString;

}

MY\_STRING my\_string\_init\_c\_string(const char\* c\_string)

{

My\_string\* pString =(My\_string\*) malloc(sizeof(My\_string));

int x = 0;

while (c\_string[x] != '\0')

{

x++;

}

pString -> capacity = x+1 ;

pString -> size = x ;

pString -> data =(char\*) malloc (sizeof(char)\*pString -> capacity);

while (c\_string[x] != '\0')

{

pString -> data[x] = c\_string[x];

x++;

}

return (MY\_STRING) pString;

}

int my\_string\_get\_capacity(MY\_STRING hMy\_string)

{

My\_string\* pString = (My\_string\*)(My\_string);

return pString -> capacity;

}

int my\_string\_get\_size(MY\_STRING hMy\_string)

{

My\_string\* pString = (My\_string\*)(My\_string);

return pString -> size ;

}

int my\_string\_compare(MY\_STRING hLeft\_string,MY\_STRING hRight\_string)

{

My\_string\* pStringL = (My\_string\*)hLeft\_string;

My\_string\* pStringR = (My\_string\*)hRight\_string;

int i=0;

for (i=0; i< pStringL ->size ;i++)

{

if ( pStringL -> data[i] > pStringR -> data[i] )

return 1;

else if (pStringL -> data[i] < pStringR -> data[i])

return -1;

else

return 0;

}

for( i=0; i< pStringR -> size ; i++)

{

if (pStringR-> data [i] > pStringL -> data[i])

return 1;

else if (pStringR -> data[i] < pStringL -> data[i])

return -1;

}

}

void my\_string\_destroy(MY\_STRING\* phMy\_string)

{

My\_string\* pString= (My\_string\*)\*phMy\_string;

free(pString -> data);

free (pString);

phMy\_string=NULL;

}