Suppose a **variable-sized string** is represented in **C** using the following typedef:

typedef struct

{

short shMaxLength; // maximum length of the string (this is the size we malloc)

short shLength; // current number of characters in the string

char \*pString; // pointer to dynamic memory containing a contiguous array

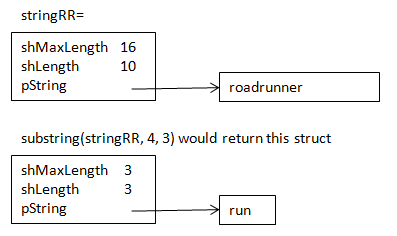
// with a size of shMaxLength. It is not zero terminated.

**IF IS ZERO TERMINATED ADD 1 WHEN MALLOCING OR REF.**

} VarString;

Show code for the C function, **VarString substring(VarString string, int iBeg, int iLgth**), which should return a substring of the specified string beginning at iBeg for a length of iLgth. Notes:

* The first character in the string will be at subscript 0. The last character will be at subscript shLength-1.
* If the request went outside the range of characters, use errExit to terminate with an appropriate message.
* It is not necessary to allocate a VarString structure using malloc since we are simply returning a structure.
* Allocate memory (use malloc) for the contiguous array of iLgth characters. If the malloc fails, use errExit to terminate with an appropriate message.
* Return the VarString structure.
* The VarString typedef is defined in "VarString.h". You do not have to include the standard C include files.
* Example:



Invoking code snippet:

#include "VarString.h"

void someFunc(VarString stringTeam)

{

VarString stringPart;

…

stringPart = substring(stringTeam, 4, 3);

…

}

#include “VarString.h”

VarString substring(Varstring string, int iBeg, int iLngth) {

VarString stringNew;

If (iBeg < 0 || iBeg+iLngth > string.shLength || iLngth < 0)

errExit(“string ranger error”);

stringNew.pString = (char \*) malloc(iLngth);

if(stringNew.pString == NULL)

errExit(“string is NULL”)

stringNew.shMaxLength = iLngth;

stringNew.shLength = iLngth;

memcpy(stringNew.pString, string.pString+iBeg, iLngth);

return stringNew;

}

C can return struct