CSC209H Worksheet: malloc and strings

1. Each time a variable is declared or memory is otherwise allocated, it is important to understand how much memory is allocated, where it will be allocated and when it will be de-allocated. Complete the table below. (Note: some of the programs allocate more than one block of memory.)

Code Fragment	**Space?**	**Where?**	**De-allocated when?**
`int main() {` `int i;` `}`	sizeof(int) 4 bytes	stack frame for `main`	when program ends
`int fun() {` `float i;` `}` `int main() {` `fun();` `}`	sizeof(float) 4 bytes 32 bits	stack frame of fun()	deallocated when the fun() returns
`int fun(char i) {` `...` `}` `int main() {` `fun('a');` `}`	sizeof(char)	stack from of fun() char i is essentially a local variable	deallocated when the fun() returns
`int main() {` `char i[10] = "hello";` `}`	sizeof(char)*10 the hello is gonna be in program data	stack frame of main when your program runs, the array is gonna be copied into the stack	deallocated when the main() returns
`int main() {` `char *i;` `}`	sizeof(char *)	stack frame of main	deallocated when the main() returns
`int main() {` `int *i;` `}`	sizeof(in *)	stack frame of main	deallocated when the main() returns
`int main() {` `char *i = "hello";` `}`	sizeof(char *) sizeof(char)*6	stack frame of main the string is gonna be in program data	deallocated when the main() returns
`int fun(int *i) {` `...` `}` `int main() {` `int i[5] = {4,5,2,5,1};` `fun(i);` `}`	sizeof(int)*5	stack frame of main	deallocated when the main() returns
`int main() {` `int *i;` `i = malloc(sizeof(int));` `}`	sizeof(int *)	allocated at the heap	when the program returns or when you call free()
`void fun(int **i) {` `*i = malloc(sizeof(int)*7);` `}` `int main() {` `int *i;` `fun(&i);` `free(i);` `}`	in main() sizeof(int *) in fun() sizeof(int **)	stack from of main stack from of main	

notice the difference between the
*i and **i
若是**i, 就dereference两次，变成i = malloc（ ）
现在是*i指向malloc（ ），**i还是指向*i

after fun() is called, int** is pointing to i*. int**is pointing to sizeof(int)*7, malloc will return a memory address and copy it to int**, so int** now is pointing to sizeof(int)*7, (int* is in main, int** is in fun) the problem here is once fun returns, int** is no longer here, the result is memory leak. Another problem is int** is pointing to malloc(sizeof(int)) the return value of malloc is void* but we assign it to int **. There is type mismatch. dereference int**[0] should be a int*.

CSC209H Worksheet: malloc and strings

2. Write a program that declares 3 strings. The first named `first` should be set to the value `"Monday"`, and be stored on the stack frame for `main`. `second` should be a string literal with the value `"Tuesday"`. `third` should have value `"Wednesday"` and be on the heap. The pointers for `second` and `third` will be in stack frame for `main`.

3. Write statements to shorten the strings to the abbreviations for the day names. For example, change `"Monday"` to `"Mon"`. Which string can not be changed in place? Why not?

4. Draw the memory model for your program.

5. Add to your program so that it declares an array `string_list` of 3 pointers to char and point the elements to `first`, `second`, and `third`, respectively. So now you have an array of strings. Where is the memory allocated for this array? Add to your picture above.

6. So far much of the allocation has happened in the function `main`. What would happen if you changed `main` to be another function `func` and then returned from it? Which parts of your structure would remain allocated? Write a new function `build_month_list` that allocates, initializes and returns an array of 3 strings with the values `"January"`, `"February"`, and `"March"`. All the strings should be mutable.