1. [25 pts] The cube in the picture is made out of smaller cubes. Its dimensions are 3x3x3 (3 cubes wide, 3 cubes high, and 3 cubes deep). Given the variables for the dimensions and the following struct which represents the smaller cubes, write the code to create and initialize a variable that represents the larger cube. You must use malloc to dynamically allocate the memory for the cube. Initialize each miniCube to by setting color to 'm' and contents to "None".

```c
int width = 3, height = 3, depth = 3;

struct miniCube {
    char color;
    char *contents;
};

int i, j, k;

struct miniCube ***cube = (struct miniCube **)malloc(sizeof(struct miniCube**) * width);

for (i = 0; i < width; i++) {
    cube[i] = (struct miniCube **)malloc(sizeof(struct miniCube*) * height);
    for (j = 0; j < width; j++) {
        cube[i][j] = (struct miniCube*)malloc(sizeof(struct miniCube) * depth);
        for (k = 0; k < depth; k++) {
            cube[i][j][k].color = 'm';
            cube[i][j][k].contents = "None";
        }
    }
}
```
2. [25pts] (Regarding command line arguments) Write a full program that prints a word to the console multiple times. Your program must read the number of repeats and the word to print out from the command line following the format in the table.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Meaning</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>-r &lt;repeats&gt;</td>
<td>Sets the number of times the program will print a line saying “Hello World” to the console.</td>
<td>3</td>
</tr>
<tr>
<td>-w &lt;word&gt;</td>
<td>Sets the word to print out</td>
<td>Hello</td>
</tr>
</tbody>
</table>

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

int repeats = 3;
char *word = "Hello";

void parseArguments(int argc, char **argv) {
    int i;
    for(i = 1; i < argc; i++) {
        if(argv[i][0] == '-') {
            switch(argv[i][1]) {
            case 'r':
                i++;
                repeats = atoi(argv[i]);
                break;
            case 'w':
                i++;
                word = argv[i];
                break;
            default:
                printf("Error \%c\n", argv[i]);
            }
        } else {
            printf("\%s is an invalid argument\n", argv[i]);
        }
    }
}

int main(int argc, char** argv) {
    parseArguments(argc, argv);
    int i;
    for(i = 0; i < repeats; i++) {
        printf("\%s\n", word);
    }
}
```