Lab assignment 2: Memory allocation

Dr Martin Porcheron

You have two weeks to complete this lab assignment—you must have it marked/signed off by a lab helper in a timetabled lab session. You must get this assignment marked by **11th February 2022** but I recommend you try and complete this as soon as possible. This assignment is worth 15 marks.

This lab task involves taking variable length input from the command line and decoding/storing the values for processing using a dynamically allocated array. All dynamically allocated memory must have its lifespan managed properly and safely using the best practices we have discussed in the lectures.

In this assignment, you will be working with *times tables* (also called multiplication tables).

I recommend you keep all your lab assignments in a new directory. If working on the lab machines, create this within your home directory for CSC371. Keep each lab assignment in a separate directory.

I have given instructions in this lab assignment for compiling code by command. I strongly encourage you to practice doing this, but you are also free to use an IDE if you wish. Using an IDE is your choice and you are responsible for configuring it.

Note: Do not copy and paste code from the PDF as it will contain non-standard characters and will not compile.

Code: Times Table Generator

Read through the following code, which generates a times table for a given number, supplied as a program argument. A copy of this code is available from the Canvas assignment page for lab assignment 2.

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

#define MAX_TABLE_SIZE 13

void generateTable(int num, int *table);
void printTable(int num, int *table);

int main(int argc , char *argv[]) {
    if (argc != 2) {
        printf("%s <num>\n where <num> is an int 0-12\n", argv[0]);
        return -1;
    }

    int num = atoi(argv[1]);
    if (num < 0 || num > 12) {
        printf("Invalid number size! Must be between 0 and 12.\n");
        return -1;
    }

    int values[MAX_TABLE_SIZE];
    generateTable(num, values);
    printTable(num, values);

    return 0;
}

void generateTable(int num, int *table) {
    int i;
    for (i = 0; i < MAX_TABLE_SIZE; i++) {
        table[i] = i * num;
    }
}

void printTable(int num, int *table) {
    printf("%-2d times table\n--------------\n", num);
    int i;
    for (i = 0; i < MAX_TABLE_SIZE; i++) {
        printf("%-2d * %-2d = %d\n", num, i, *(table+i));
    }
}
```

Task 1: Heap memory (3 marks)

In this exercise, you are going to modify the above file to use heap memory.

1. Download a copy of the source code above from the Canvas assignment page for lab assignment 2 and name this file *task1.c*.
2. Run the code and examine how it works. You should make sure you understand it fully.
3. Modify the `main` function by swapping it from using stack memory to heap memory for the storage of the generated times table.
    - To do this, you will need to replace the line `int values[MAX_TABLE_SIZE];`
    - You must appropriately `free` memory at then end of your program too
4. From the command line compile and link your program with the GCC compiler using:

    ```
    $ gcc task1.c -o task1
    ```

Expected output

If you run your program using the command:

```
$ ./task1 12
```

...you should get the output:

```
12 times table
--------------
12 * 0  = 0
12 * 1  = 12
12 * 2  = 24
12 * 3  = 36
12 * 4  = 48
12 * 5  = 60
12 * 6  = 72
12 * 7  = 84
12 * 8  = 96
12 * 9  = 108
12 * 10 = 120
12 * 11 = 132
12 * 12 = 144
```

Task 2: Multi-dimensional array (7 marks)

In this exercise you are required to modify the program you previously generated in task 1 to generate the times tables for the numbers 0–12 automatically (i.e. your code will generate the 0 times table, then the 1 times table, then the 2 times table, and so on up to and including the 12 times table).

1. Copy your task 1 file and name this new version *task2.c*.
2. Create two new files, *timestables.h* and *timestables.c*. You should move all functions and function prototypes except the `main` function from *task2.c* into these two new files. You should add the necessary `#includes`.
   - You should move `#define MAX_TABLE_SIZE 13` to *timestables.h*.
   - You must add header guards to your header file.
   - You should define a macro in *timestables.h*, `MAX_TIMES_TABLE`, that has value 12.
3. Add this code at the bottom of your *timestables.c*:

   ```
   void printTables(int **tables) {
       int i;
       for (i = 0; i <= MAX_TIMES_TABLE; i++) {
           printTable(i, tables[i]);
       }
   }
   ```

   - Spend some time thinking through what this function is doing. Add a comment that explains this.
   - You also need to add the appropriate function prototype to *timestables.h*.
4. Modify the `main` function in *task2.c* by removing the code that reads and validates the program argument.

   Instead, create an array for pointers on the stack. Your array must be large enough to generate the times tables 0, 1, 2, ..., 11, and 12 (i.e. it will generate 13 sets of times tables).
5. Create a `for` loop that uses `malloc` inside it to allocate enough space for each times table in your `main` function in *task2.c*.

   Store the pointer from `malloc` in the array stored in stack memory at the correct index (i.e. at index 0 store the pointer to the location in heap memory allocated for the 0 times table, at index 1 store the pointer for the location in heap memory allocated for the 1 times table...).
6. Modify the `main` function to call the `printTables` function.
7. You may need to adjust how you `free` memory to be in an additional loop (i.e., to free each pointer!)
8. From the command line compile your program to object files with the GCC compiler using:

   ```
   $ gcc -c task2.c -o task2.o
   $ gcc -c timestables.c -o timestables.o
   ```
9. Link your two files:

   ```
   $ gcc timestables.o task2.o -o task2
   ```

Expected output

If your run your program using the command:

```
$ ./task2
```

...you should get the output:

```
0  times table
-------------
0  * 0  = 0
0  * 1  = 0
0  * 2  = 0
0  * 3  = 0
0  * 4  = 0
0  * 5  = 0
0  * 6  = 0
0  * 7  = 0
0  * 8  = 0
0  * 9  = 0
0  * 10 = 0
0  * 11 = 0
0  * 12 = 0

1  times table
-------------
1  * 0  = 0
1  * 1  = 1
1  * 2  = 2
1  * 3  = 3
1  * 4  = 4
1  * 5  = 5
1  * 6  = 6
1  * 7  = 7
1  * 8  = 8
1  * 9  = 9
1  * 10 = 10
1  * 11 = 11
1  * 12 = 12

2  times table
-------------
2  * 0  = 0
2  * 1  = 2
2  * 2  = 4
2  * 3  = 6
2  * 4  = 8
2  * 5  = 10
2  * 6  = 12
2  * 7  = 14
2  * 8  = 16
2  * 9  = 18
2  * 10 = 20
2  * 11 = 22
2  * 12 = 24

3  times table
-------------
3  * 0  = 0
3  * 1  = 3
3  * 2  = 6
3  * 3  = 9
3  * 4  = 12
3  * 5  = 15
3  * 6  = 18
3  * 7  = 21
```

```
3  * 8  = 24
3  * 9  = 27
3  * 10 = 30
3  * 11 = 33
3  * 12 = 36

4  times table
--------------
4  * 0  = 0
4  * 1  = 4
4  * 2  = 8
4  * 3  = 12
4  * 4  = 16
4  * 5  = 20
4  * 6  = 24
4  * 7  = 28
4  * 8  = 32
4  * 9  = 36
4  * 10 = 40
4  * 11 = 44
4  * 12 = 48

5  times table
--------------
5  * 0  = 0
5  * 1  = 5
5  * 2  = 10
5  * 3  = 15
5  * 4  = 20
5  * 5  = 25
5  * 6  = 30
5  * 7  = 35
5  * 8  = 40
5  * 9  = 45
5  * 10 = 50
5  * 11 = 55
5  * 12 = 60

6  times table
--------------
6  * 0  = 0
6  * 1  = 6
6  * 2  = 12
6  * 3  = 18
6  * 4  = 24
6  * 5  = 30
6  * 6  = 36
6  * 7  = 42
6  * 8  = 48
6  * 9  = 54
6  * 10 = 60
6  * 11 = 66
6  * 12 = 72

7  times table
--------------
7  * 0  = 0
7  * 1  = 7
```

```
7  * 2  = 14
7  * 3  = 21
7  * 4  = 28
7  * 5  = 35
7  * 6  = 42
7  * 7  = 49
7  * 8  = 56
7  * 9  = 63
7  * 10 = 70
7  * 11 = 77
7  * 12 = 84

8  times table
--------------
8  * 0  = 0
8  * 1  = 8
8  * 2  = 16
8  * 3  = 24
8  * 4  = 32
8  * 5  = 40
8  * 6  = 48
8  * 7  = 56
8  * 8  = 64
8  * 9  = 72
8  * 10 = 80
8  * 11 = 88
8  * 12 = 96

9  times table
--------------
9  * 0  = 0
9  * 1  = 9
9  * 2  = 18
9  * 3  = 27
9  * 4  = 36
9  * 5  = 45
9  * 6  = 54
9  * 7  = 63
9  * 8  = 72
9  * 9  = 81
9  * 10 = 90
9  * 11 = 99
9  * 12 = 108

10 times table
--------------
10 * 0  = 0
10 * 1  = 10
10 * 2  = 20
10 * 3  = 30
10 * 4  = 40
10 * 5  = 50
10 * 6  = 60
10 * 7  = 70
10 * 8  = 80
10 * 9  = 90
10 * 10 = 100
10 * 11 = 110
```

```
10 * 12 = 120

11 times table
--------------
11 * 0  = 0
11 * 1  = 11
11 * 2  = 22
11 * 3  = 33
11 * 4  = 44
11 * 5  = 55
11 * 6  = 66
11 * 7  = 77
11 * 8  = 88
11 * 9  = 99
11 * 10 = 110
11 * 11 = 121
11 * 12 = 132

12 times table
--------------
12 * 0  = 0
12 * 1  = 12
12 * 2  = 24
12 * 3  = 36
12 * 4  = 48
12 * 5  = 60
12 * 6  = 72
12 * 7  = 84
12 * 8  = 96
12 * 9  = 108
12 * 10 = 120
12 * 11 = 132
12 * 12 = 144
```

Task 3: Pointless means (5 marks)

In this exercise, you will have to calculate the mean for each times table. You do not need to modify *task2.c*.

1. Create two new files, *arrays.h* and *arrays.c*. Add appropriate header guards to your header file.
2. Create a function named `mean` in *arrays.c* that takes two arguments: an `integer` named `length` and an integer pointer (`int *`) named `arr`. The function should return a `double`. You should add the prototype for the function to *arrays.h*.
3. Write the code that calculates the mean for an array provided inside the `mean` function. The mean is calculated by going over the array passed in, adding all values together and dividing by the number of values provided.
4. Modify `printTable` in *timestables.c* to add the following code to the bottom of the function:

   ```
   double meanValue = mean(MAX_TABLE_SIZE, table);
   printf("The mean for this table is %f\n\n", meanValue);
   ```

   You will need to `#include` the appropriate file to make your code compilable.
5. Compile the modified part of your program (we don't need to recompile task2.c as we have not changed it):

   ```
   $ gcc -c timestables.c -o timestables.o
   $ gcc -c arrays.c -o arrays.o
   ```
6. Relink the modified and new code to your code from Task 2:

   ```
   $ gcc arrays.o timestables.o task2.o -o task3
   ```

Expected output

If your run your program using the command:

```
$ ./task3
```

...you should get the output:

```
0  times table
-------------
0  * 0  = 0
0  * 1  = 0
0  * 2  = 0
0  * 3  = 0
0  * 4  = 0
0  * 5  = 0
0  * 6  = 0
0  * 7  = 0
0  * 8  = 0
0  * 9  = 0
0  * 10 = 0
0  * 11 = 0
0  * 12 = 0

The mean for this table is 0.000000

1  times table
-------------
1  * 0  = 0
1  * 1  = 1
1  * 2  = 2
```

```
1  * 3  = 3
1  * 4  = 4
1  * 5  = 5
1  * 6  = 6
1  * 7  = 7
1  * 8  = 8
1  * 9  = 9
1  * 10 = 10
1  * 11 = 11
1  * 12 = 12

The mean for this table is 6.000000

2  times table
--------------
2  * 0  = 0
2  * 1  = 2
2  * 2  = 4
2  * 3  = 6
2  * 4  = 8
2  * 5  = 10
2  * 6  = 12
2  * 7  = 14
2  * 8  = 16
2  * 9  = 18
2  * 10 = 20
2  * 11 = 22
2  * 12 = 24

The mean for this table is 12.000000

3  times table
--------------
3  * 0  = 0
3  * 1  = 3
3  * 2  = 6
3  * 3  = 9
3  * 4  = 12
3  * 5  = 15
3  * 6  = 18
3  * 7  = 21
3  * 8  = 24
3  * 9  = 27
3  * 10 = 30
3  * 11 = 33
3  * 12 = 36

The mean for this table is 18.000000

4  times table
--------------
4  * 0  = 0
4  * 1  = 4
4  * 2  = 8
4  * 3  = 12
4  * 4  = 16
4  * 5  = 20
4  * 6  = 24
```

```
4  * 7  = 28
4  * 8  = 32
4  * 9  = 36
4  * 10 = 40
4  * 11 = 44
4  * 12 = 48

The mean for this table is 24.000000

5  times table
--------------
5  * 0  = 0
5  * 1  = 5
5  * 2  = 10
5  * 3  = 15
5  * 4  = 20
5  * 5  = 25
5  * 6  = 30
5  * 7  = 35
5  * 8  = 40
5  * 9  = 45
5  * 10 = 50
5  * 11 = 55
5  * 12 = 60

The mean for this table is 30.000000

6  times table
--------------
6  * 0  = 0
6  * 1  = 6
6  * 2  = 12
6  * 3  = 18
6  * 4  = 24
6  * 5  = 30
6  * 6  = 36
6  * 7  = 42
6  * 8  = 48
6  * 9  = 54
6  * 10 = 60
6  * 11 = 66
6  * 12 = 72

The mean for this table is 36.000000

7  times table
--------------
7  * 0  = 0
7  * 1  = 7
7  * 2  = 14
7  * 3  = 21
7  * 4  = 28
7  * 5  = 35
7  * 6  = 42
7  * 7  = 49
7  * 8  = 56
7  * 9  = 63
7  * 10 = 70
```

```
7  * 11 = 77
7  * 12 = 84

The mean for this table is 42.000000

8  times table
--------------
8  * 0  = 0
8  * 1  = 8
8  * 2  = 16
8  * 3  = 24
8  * 4  = 32
8  * 5  = 40
8  * 6  = 48
8  * 7  = 56
8  * 8  = 64
8  * 9  = 72
8  * 10 = 80
8  * 11 = 88
8  * 12 = 96

The mean for this table is 48.000000

9  times table
--------------
9  * 0  = 0
9  * 1  = 9
9  * 2  = 18
9  * 3  = 27
9  * 4  = 36
9  * 5  = 45
9  * 6  = 54
9  * 7  = 63
9  * 8  = 72
9  * 9  = 81
9  * 10 = 90
9  * 11 = 99
9  * 12 = 108

The mean for this table is 54.000000

10 times table
--------------
10 * 0  = 0
10 * 1  = 10
10 * 2  = 20
10 * 3  = 30
10 * 4  = 40
10 * 5  = 50
10 * 6  = 60
10 * 7  = 70
10 * 8  = 80
10 * 9  = 90
10 * 10 = 100
10 * 11 = 110
10 * 12 = 120

The mean for this table is 60.000000
```

```
11 times table
--------------
11 * 0  = 0
11 * 1  = 11
11 * 2  = 22
11 * 3  = 33
11 * 4  = 44
11 * 5  = 55
11 * 6  = 66
11 * 7  = 77
11 * 8  = 88
11 * 9  = 99
11 * 10 = 110
11 * 11 = 121
11 * 12 = 132

The mean for this table is 66.000000

12 times table
--------------
12 * 0  = 0
12 * 1  = 12
12 * 2  = 24
12 * 3  = 36
12 * 4  = 48
12 * 5  = 60
12 * 6  = 72
12 * 7  = 84
12 * 8  = 96
12 * 9  = 108
12 * 10 = 120
12 * 11 = 132
12 * 12 = 144

The mean for this table is 72.000000
```