CSSE 132 – Introduction to Computer Systems
Rose-Hulman Institute of Technology
Computer Science and Software Engineering Department

Stack/Heap Practice

Name:_____ Mail:_____

1. Fill the assembly code into the blank below to complete an *Add* function. The function in C is also shown below.

```c
void Add(int * a, int * b)
{
        int c = *a + *b;
        *a = c;
}
```

```
Add:
        sub sp, sp, #12 @ Reserve stack space for local
           variables: a, b, c
        str r0, ________ @ store input argument (a) on the
           stack
        str ________, ________ @ store input argument (b)
           on the stack
        ldr ________, ________ @ load the value inside of
           pointer (a) into r0
        ldr ________, ________ @ load the value inside of
           pointer (b) into r1
        add r0, r0, r1
        str r0, ________ @ put the add result into (c)
        ldr r1, ________ @ load the pointer (a) into r1
        ldr r2, ________ @ load the result (c) into r2
        str r2, [r1] @ store result as the new value of
           pointer (a)
        add sp, sp, ________ @ Release stack
        bx lr @ return
```

Solution:

```
1  Add:
2          sub sp, sp, #12 @ Reserve stack space for local
              variables: a, b, c
3          str r0, [sp, #4] @ store input argument (a) on the
              stack
4          str r1, [sp, #8] @ store input argument (b) on the
              stack
5          ldr r0, [r0] @ load the value inside of pointer (a)
               into r0
6          ldr r1, [r1] @ load the value inside of pointer (b)
               into r1
7          add r0, r0, r1
8          str r0, [sp] @ put the add result into (c)
9          ldr r1, [sp, #4] @ load the pointer (a) into r1
10         ldr r2, [sp] @ load the result (c) into r2
11         str r2, [r1] @ store result as the new value of
              pointer (a)
12         add sp, sp, #12 @ Release stack
13         bx lr @ return
```

2. Create 2D array on the heap with the minimal `malloc` calling. *NOTE: the return pointer must be the type of **int ****, so that we can use retrieve an element by using **2dArray[i][j]**.*

```
#include <malloc.h>

int** My2DAlloc(int rows, int cols) {









        return 2dArray;
}
```

Traditional Solution:

```c
#include <malloc.h>

int** My2DAlloc(int rows, int cols) {

        int** 2dArray = (int**)malloc(rows * sizeof(int*));

        int k;
        for(k=0;k<rows;k++){
                2dArray[k] = (int *)malloc(cols*sizeof(int)
                   );
        }
        return 2dArray;
}
```

Best Solution:

```c
#include <malloc.h>

int** My2DAlloc(int rows, int cols) {

        int header = rows * sizeof(int*);
        int data = rows * cols * sizeof(int);
        int** 2dArray = (int**)malloc(header + data);
        int* buf = (int*)(2dArray + rows);
        int k;
        for(k=0;k<rows;k++){
                2dArray[k] = buf + k*cols;
        }
        return 2dArray;
}
```