Brian Gunnarson

CIS 330 MIDTERM

1.) a.) double i = (3.1 * (3.0) * 3.0) * 5.0+1;

b.) A= 0x8040 (pointer is 8 bytes; add 5 of them; 8*5=40; 0x8000+40 = 0x8040)

c.) ./a.out Doctor Who?

2.) a.) D

b.) F

c.) D

3.)
```
void swap_arr(int** arr, int m) {
    for (int i=0; i<m; i++) {          // loop through rows
        for (int j=0; j<m; j++) {      // loop through cols
            int tmp = arr[i][j];       // store temporary value
            arr[i][j] = arr[j][i];     // swap arr[i][j] with arr[j][i]
            arr[j][i] = tmp;           // replace arr[j][i] with temp value
        }
    }
}
```

4.) a.)
```
void init_2d(int**** array_2d, int first_dim, int second_dim) {
    *array_2d = (int***) malloc(sizeof(int**) * first_dim);
    for (int i=0; i < first_dim; i++) {
        array_2d[i] = (int**) malloc(sizeof(int*) * second_dim);
    }
}
```

b.)
```
void free_2d(int*** array_2d, int first_dim, int second_dim) {
    for (int i=0; i< first_dim; i++) {
        free(array_2d[i]);
    }
    free(array_2d);
}
```

5.) a.)

```
void init_2d_grid(point_t **** my_grid, int rows, int cols) {
    *my_grid = (point_t ***) malloc(sizeof(point_t **) * rows);
    for (int i=0; i<rows; i++) {
        my_grid[i] = (point_t **) malloc(sizeof(point_t *) * cols);
        for (int j=0; j<cols; j++) {
            my_grid[i][j] = (point_t*) malloc(sizeof(point_t));
            my_grid[i][j]->val = rand() % 10;
            my_grid[i][j]->x = (j+1) - 1;
            my_grid[i][j]->y = rows - 1 - i;
        }
    }
}
```

b)

```
int sum_vals(point_t *** my_grid, int rows, int cols, int a, int b) {
    int sum = 0;

    for (int i=0; i<a+1; i++) {
        sum += my_grid[rows-1][i]->val;
    }
    for (int j=1; j<b+1; j++) {
        sum += my_grid[rows-1-j][a]->val;
    }
    return sum;
}
```

6.)
```
int * convert_1d(int ** array_2d, int rows, int cols) {
    int * array_1d = (int *) malloc(sizeof(int) * rows * cols);
    int count = 0;
    for (int i=0; i<cols; i++) {
        for (int j=0; j<rows; j++) {
            array_1d[count] = array_2d[j][i];
            count++;
        }
    }
    return array_1d;
}
```

Extra Credit

(1) Preprocessor: transforms source code by including header files and expanding #defines. It takes source code as input and outputs preprocessed source code.

(2) Compiler: converts source code into assembly code using 3 stages:
a) Front end b.) Middle end c) Back end

(3) Assembler: converts the assembly code produced by the compiler into a binary object file.

(4) Linker: merges object files into a single executable and resolves external references