Lab 12

ID1303: Introduction to Programming

1. Consider the following definition of a structure that can store a vector.

```
struct vector
{
   double *v;
   unsigned int d;
};
```

   where d is the dimension of the vector.

   (a) Create a variable of type vector; accept the value of d from the user and dynamically allocate memory for v to store d entries.

   (b) Write a function called initializeVector that accepts (i) the address of a struct vector variable, (ii) the value of d, (iii) a double value a, and dynamically allocates memory for v to store d entries, initializing each entry to the value a.

   (c) Define an enum variable called BasicDataType with the values Char, Int, Float, Double. Change the declaration type of v from double * to void* and write a function called memoryAssign that accepts (i) the address of a struct vector variable, (ii) the value of d, (iii) a BasicDataType value t, and dynamically assigns memory for v to store d entries whose data type is t.

2. Consider the following declarations:

```
double **matrix;
unsigned int m,n;
```

   Accept the value of m, n from the user and dynamically allocate memory for $matrix$ to store a rectangular matrix of size $m \times n$.

3. (a) Suppose that $f(x, y)$ is a C function with two arguments that stores its result in x. Write a function iterateFunction, that accepts (i) the name of such a function, (ii) a positive integer d, (iii) address of x and the value of y, and computes the iterated function $f^d(x, y)$ which is defined recursively as follows: $f^1(x, y) = f(x, y)$; $f^d(x, y) =$

$f(f^{d-1}(x,y),y)$. Test iterateFunction on (i) pow(x,y) function from math.h, (ii) the strcat function (or on your implementation from Lab 9); calling iterateFunction with strcat, 3 and two strings YA and HOO, should produce YAHOOHOOHOO.

(b) Write a function swapFirstInversion that accepts (i) a double array num[], (ii) a positive integer n, and finds the first index $i \leq n-2$ (if any) such that $num[i] > num[i+1]$ and swaps num[i] with num[i+1]. For example, if the array has elements $3, 5, 6, 0, 2$ calling the function with $n = 3$ will swap num[2] with num[3] to give $3, 5, 0, 6, 2$, while calling the function with $n = 2$ will not have any effect (on the initial array).

Test iterateFunction from part (a) on swapFirstInversion with n as the length of the array and $d = n(n-1)/2$.