GNG1106 – Fundamentals of Engineering Computation
Practice Questions

Short Answer Questions

1) Provide the output of the following C code in the adjacent box?

```
void main()
{
    int a = 4;
    int b = 3;

    if(a < 3)
        b = 4;
    else
        a = 10;

    if(b == 3)
        b = 5;
    else
        a = 11;

    if(b > 4)
        b = 6;
    else
        a = 12;

    printf("a = %d b = %d\n",a, b);
}
```

Console

2) Assuming a function with the header "**`void updateValue(double *wPtr)`**". Declare an appropriate variable and a function call used to initialize the variable.

3) Given the following structure definition and declaration:

```
typedef struct
{
   char desc[20];
   int num;
   double cost;
} ITEM;
ITEM inventory[25];
```

   a. What is wrong with the following instruction? Correct the instruction.

```
ITEM[1].cost = 92.32;
```

   b. What is wrong with the following instruction? Correct the instruction.

```
inventory.cost[10] = 32.12;
```

4) Write a C statement that assigns the address of the floating-point variable `mass` to a pointer called `mPtr`? (1 point)

5) What is the effect of the following statements to the file system (assume no error occurs during execution):

```
FILE *fPtr;
fPtr = fopen("exam.txt","w");
fclose(fPtr);
```

6) Given that the ASCII file `exam.txt` contains the following contents:

```
First line of text.
Second line of text.
Third line of text.
```

What will the character array `string` contain after the execution of the following instructions?

```
FILE *fPtr;
char string[10];
fPtr = fopen("exam.txt","r");
fgets(string, 10, fPtr);
```

Please show ALL characters loaded into the array.

7) Provide the output of the following C code in the adjacent box. (2 points)

```
main()
{
      int x=50, y=70;
      interchange1(x,y);
      printf("First : x=%d y=%d \n",x,y);
      interchange2(&x,&y);
      printf("Now: x=%d y=%d",x,y);
   }

   void interchange1 (int x, int y)
   {
      int z = x;
      x=y;
      y=z;
   }

   void interchange2 (int *x, int *y)
   {
      int z = *x;
      *x=*y;
      *y=z;
   }
```

Console

8) Write C instructions that prompts the user for a file name and copy the contents of the file to another file, named "Copy.txt". Print "File copied …" if the copy process is successful, and "File does not exist" otherwise. Be sure **not** to leave any open files after the code has executed and the file data should **not** be altered or deleted (Also make sure that you make all required declarations).

C Code

9) In C, `NULL` is
   a) a character written as `'\0'` that terminates any string in C
   b) the address zero that does not reference anything
   c) both a) and b) / a) et b)
   d) neither a) nor b) / ni a) ni b)

For the following Questions 10 to 12, consider the following definition. Assume that integers occupy 4 bytes and double 8 bytes.

```
typedef struct
{
          int num;
          double realNum;
          char str[100];
} ASTRUCT;
```

10) The declaration "`ASTRUCT var;`"
   a) Reserves 112 bytes for `var`,
   b) Reserves 4 bytes for `var`,
   c) Reserves 100 bytes for `var`,
   d) None of the above answers.

11) Given the declaration `ASTRUCT var;` and assume that `var` has been initialized.
   a) `var->num` gives the integer value stored in the member num,
   b) `var.str` gives the address of where the array `str` is stored,
   c) `var.num + var.realNum` gives an integer value
   d) None of the above answers.

12) Given the declaration `ASTRUCT var;` and assume that `var` has been initialized.
   a) The function call `func(var)` passes an address to `func`,
   b) The function call `func(var)` allows the function `func` to update all members of `var`.
   c) The function call `func(var)` passes the complete contents of `var` to `func`.
   d) a) and b) are true.

Computer Algorithms

1) Complete the function `bubleSortCharReverse` to sort a character array in reverse order. Improve its efficiency by incorporating the following logic used with the replacement sort algorithm:
   - For the replacement sort algorithm, after each pass, the sub-array being searched was reduced by 1 element given that the last element is sorted.
   - This is the case with the bubble sort algorithm, that is, after each pass, the last element contains a sorted algorithm.
   - Thus, with the bubble sort algorithm, it is possible after each pass, to reduce the subarray being sorted by one element.

```
/*-----------------------------------------------
Function: bubbleSortCharReverse
Parameters:
   n - number of characters in the array
   array - pointer to an array of characters
Description:  A function that sorts the array
   of characters.  The function is efficient.
-----------------------------------------------*/
void bubbleSortCharReverse(int n, char array[])
{ ... }
```

2) Consider a 2D character array that contains a string in each row. Complete the following function that uses the replacement sort algorithm to sort the strings in order. You will need to use string library functions to compare and copy strings.

```
/*-------------------------------------------------
Function: sort_repl_string
Parameters:
    num - number of rows in the array, i.e. number of strings
    num_char - max number of chars in rows (size of column),
               i.e. max size of string
    strArr - reference to a 2D array that contains strings to sort.
Description: Sorts the referenced array of strings
             using the replacement sort algorithm.
-------------------------------------------------*/
void sort_repl_string(int num, int num_char, char strArr[][num_char])
{ ... }
```

3) Complete the following function that uses the bubble sort algorithm to sort an array of real values. Given that equality of real values is difficult, used a symbolic constant EPSILSON (0.001) so that values are considered equal if they have a different no greater than EPSILON.

```
#define EPSILON 0.001
/*-----------------------------------------------
Function: bubbleSortReaValues
Parameters:
   n - number of values in the array
   array - pointer to an array of real values
Description:  A function that sorts the array
   of real values. Considered equal if values have
   difference less than EPSILON.
-----------------------------------------------*/
void bubbleSortRealValues(int n, double array[])
{ ... }
```

4) Consider a 2D character array that contains a string in each row. Complete the following function that uses binary search for finding a string in the array. You will need to use string library functions to compare strings.

```
/*-------------------------------------------
Function: binarySearchString
Parameters:
    key - the value to search in the array
    num - number of rows in the array, i.e. number of strings
    num_char - max number of chars in rows, i.e. max size of string
    strArr - reference to a 2D array that contains strings to sort.
Return Value: Position (index) of key in array
              or -1 if key not found.
Description: Returns the position of the key
             found (i.e. row where key is found)
             or -1 if the key is not found in the array.
-------------------------------------------*/
int binarySearchString(char *key, int num, int num_char, char arr[][num_char])
{ ... }
```

5) Complete the following function that searches for all elements in an array that equals a `key`. The function returns the number of times the `key` was found in the array, and stores the indexes of the matching element in the referenced array `found`.

```
/*-----------------------------------------
Function: linearSearch
Parameters:
    key - the value to search in the array
    num - the number of elements in both arrays (arr and found)
    arr - reference to an array of integer
          values
    found - reference to array for storing the indexes of elements
            that match the key
Return Value: Number of times the key was found.
Description: Searches for all occurences of the key in the
             array.
-----------------------------------------*/
int linearSearchAll(int key, int num, int arr[], int found[])
{ ... }
```

6) Complete the following recursive function to compute the sum of integer numbers until N.

```
/*-----------------------------------------------------
Function: sumN
Parameters:
    integer:  integer value
Description: Recursive function for computing the
             sum from 1 to N (positive integer).
-----------------------------------------------------*/
int sumN(int integer)
{ ... }
```

7) Complete the following recursive function that computes the power of x^b where b is an integer and x a real number.

```
/*-------------------------------------------------
Function: powerRecursive
Parameters:
    x:  real value
    b:  positive integer value
Description: Recursive function for computing x
             to the power of b.
-------------------------------------------------*/
double powerRecursive(double x, int b)
{ ... }
```

8) Complete the following recursive function that reverses a string in a character array, that is, "abcdefg" becomes "gfedcba" (hint: swap first and last characters until the middle is reached).

```
/*-------------------------------------------------
Function: reverseString
Parameters:
    n:  number of chars to reverse
    str: reference to string to reverse
Description: Recursive function for reversing the
             characters in a string.
-------------------------------------------------*/
void reverseString(int n, char str[])
{ ... }
```

Numerical Methods

1) If x^n = c, then $x^n - c = 0$ and the n'th root of c, $\sqrt[n]{c}$, is a zero of the second equation (note that *n* does not need to be an integer). Using the bisection method for root finding complete the following C function that finds the *n*'th root of *c*. Use the interval 0 to *c* for searching the root and compute the root to the 6th decimal place (to terminate the root finding process use the compare the difference between x_U and x_L to 10^{-7} to get the desired precision for the root). Use the C standard function `pow` to compute x^n.

```
/*-------------------------------------------------------------
Function: nthroot
Parameters
    n - real value for finding the nth root
    c - real value to find nth root of c
Returns:  The nth root of c
Description: Finds the nth root of c by finding the root
             of the function f(x)= x^n - c using the bisection
             root finding method with the interval from 0
             to c.   EPSILON is set to 10e-6 and termination
             of the root finding is done when the search interval
             has been reduced within ESPSILON.
-------------------------------------------------------------*/
#define EPSILON 10e-7
double nthroot(double n, double c)
{ ... }
```

2) The saturation concentration of dissolved oxygen in freshwater can be calculated with the equation (APHA, 1992)

$$\ln(o_{sf}) = -139.34411 + \frac{1.575701\times10^{5}}{T_a} - \frac{6.642308\times10^{7}}{T_a^2} + \frac{1.2438\times10^{10}}{T_a^3} - \frac{8.621949\times10^{11}}{T_a^4}$$

where o_{sf} = the saturation concentration of dissolved oxygen in freshwater at 1 atm (mg/L) and T_a = absolute temperature (K). Remember that $T_a = T + 273.15$, where T = temperature (^{0}C). According to this equation, saturation decreases with increasing temperature. For typical natural waters in temperate climates, the equation can be used to determine that oxygen concentration ranges from 14.621 mg/L at 0◦C to 6.413 mg/L at 40 ◦C (note range for water temperature).

(a) Modify the above formula so that a root finding method can be used, that is, define a function $f(T_a)$ for which the root gives the temperature for a given saturation concentration.

(b) Complete the following function that returns the temperature of the water in ^{0}C given a value of oxygen saturation concentration. Apply the bisection method in your solution. Return -9999 if the saturation concentration argument is invalid. Create a separate function `f_ta` to calculate the value of $f(T_a)$ for given values of T_a and o_{sf}.

```
/*-------------------------------------------------------------
Function: getSatTemp
Parameters:
    osf - saturation temperature of dissolved oxygen in fresh water
Returns:  The freshwater temperature
Description: Finds the temperature using a root finding method
             root finding method with the interval from 0
             to 40 degrees C.   EPSILON is set to 10e-6 and termination
             of the root finding is done when the search interval
             has been reduced within ESPSILON.
-------------------------------------------------------------------*/
#define EPSILON 10e-7
#define TEMP_L 0.0 // Degrees C
#define TEMP_U 40.0 // Degrees C
#define TCONV 273.15 // To convert between C and K
double getSatTemp(double osf)
{ ... }
```

3) Given the following equation $x^2 \cos(x) = 5$.
   (a) Define the function *f(x)* whose roots gives a solution for the above equation.
   (b) Complete the following function that uses the incremental search method to find all solutions for *x* (i.e. all roots of *f(x)*) for a given interval of x. You may use additional functions in your solution.

```
/*-------------------------------------------------------------
Function: findAllRoots
Parameters:
    start, end:  start and end of interval (x values) for searching roots
    roots - reference to array for storing roots
Returns:
    The number of roots found.
Description: Find the roots between the interval for the
             the function f(x) = ... (from part a).
-------------------------------------------------------------*/
int findAllRoots(double start, double end, double roots[])
{ ... }
```

4) If water is drained from a vertical cylindrical tank by opening a valve at the base, the water will flow fast when the tank is full and slow down as it continues to drain. As it turns out, the rate at which the water level drops is:

$$\frac{dy}{dt} = -k\sqrt{y}$$

where *k* is a constant depending on the shape of the hole and the cross-sectional area of the tank and drain hole. The depth of the water *y* is measured in meters and the time *t* in minutes.
   (a) Using Euler's method, write the difference equation for the above equation to solve for *y(t)*. Define a value for the Euler time step to obtain an accuracy $1/100^{th}$ of a minute.
   (b) Using results from (a), complete the following equation that finds the time at which the tank is totally drained given, the constant k and the initial depth for y. Hint: solve for y until it reaches 0 (or becomes negative) and use the time *t* at that point as the result..

```
/*-------------------------------------------------------------
Function: calculateDrainTime
Parameters:
    k - Constant k that depends on shape of drain hole and
        cross-sectional area of tank.
    yInit - Initial depth of the liquid in the tank.
Return Value:
    The time it takes for all the liquid to drain from the tank.
Description: Using Euler's method, computes the time it takes
             to drain all the liquid from the cylindrical
             tank.
-------------------------------------------------------------*/
double calculateDrainTime(double k, double yInit)
{ ... }
```

5) Assuming that drag is proportional to the square of velocity, we can model the velocity of a falling object like a parachutist with the following differential equation:

$$\frac{dv}{dt} = g - \frac{c_d}{m} v^2$$

where v is velocity (m/s), t = time (s), g is the acceleration due to gravity (9.81 m/s^2), c_d = a second-order drag coefficient (kg/m), and m = mass (kg). Consider that the distance from where the object started its fall (point of release) is given by a simple differential equation:

$$\frac{dx}{dt} = -v$$

(a) Using Euler's method develop the 2 difference equations from the above differential equations that solves for both velocity *v(t)* and distance *x(t)* from point of release. Provide the initial conditions and a suitable value for the Euler time step.

(b) Using the results from (a), complete the following function that returns the time it takes for the object to reach the ground given the initial height of the object, its weight *m*, the drag coefficient c_d. Note that the time for the object to reach the ground is the time it takes for *x(t)* to reach 0.0.
(Note that this problem illustrates how to solve a system of differential equations).

```
/*-------------------------------------------------------------
Function: calculateDropTime
Parameters:
    height - Initial height at point of release in meters
    weight - Weight of object in kg.
    drag - drag coefficient cd in km/m.
Return Value:
    The time it takes for the object to reach the ground.
Description: Using Euler's method, computes the time it takes
             for the falling object to reach the ground.
-------------------------------------------------------------*/
#define DELTA_T 0.01  // in seconds
double calculateDropTime(double height, double weight, double drag)
{ ... }
```

6) The logistic model is used to simulate population as in

$$\frac{dp}{dt} = k_{gm}\left(1 - \frac{p}{p_{\max}}\right)p$$

where p = population (million people), k_{gm} = the maximum growth rate under unlimited conditions (year^{-1}), and $p_{\max}$ = the carrying capacity (10^6 million people).

(a) Using Euler's method develop the difference equation from the above differential equation that solves the change of population. Provide the initial conditions.

(b) Using the above results, complete the following function, that fills the time/population arrays to simulate how the population grows between two given years and suitable for plotting. Use the size of the arrays to determine the time step.

```
/*-------------------------------------------------------------
Function: calculatePopulationEuler
Parameters:
    popInit – initial population at year y1
    kgm - growth rate
    pmax - carrying capacity
    y1, y2 - range of years to compute population increase
    n - number of points in array, i.e. number of columns
    popPoints – 2D array that contains points of the time/population
                Row T_IX contains time values
                Row POP_IX contains population values.
Description: Computes the change in population from y1
             to y2 using Euler's method to solve differential
             equation.
-------------------------------------------------------------*/
#define T_IX 0  // Row for time values in popPoints
#define POP_IX 1 // Row for population values in popPoints
void calculatePopulationEuler(double popInit, double kgm,  double pmax,
                              double y1, double y2,
                              int n, double popPoints[][n])
{ ... }
```

7) Consider the following four definite integrals.

$$C = \int_1^2 \left(\frac{x+1}{x} \right)^2 dx$$

$$C = \int_{-3}^5 \left(4x-3\right)^3 dx$$

$$C = \int_0^3 x^2 e^x dx$$

$$C = \int_0^1 15^{2x} dx$$

Develop a C function that provides the results of any of the four integrals using the trapezoidal rule. The limits of integration are provided as parameters while another parameter is used to identify the integral to evaluate (a value of one of the symbolic constants `I1` to `I4`). Generalise the C integration function by creating a second C function that evaluates the function of x, *f(x)*, being integrated (for example in the second case $f(x) = (4x\text{-}3)^3$). Use a symbolic constant to define the number of steps used for the integration (e.g. `#define num_steps 20`). Recall the general result of applying the trapezoidal rule:

$$\int_a^b f(x)dx = \frac{h}{2}\left[f(x_0) + \left[2\sum_{i=1}^{n-1} f(x_i) \right] + f(x_n) \right]$$

```
// Define symbolic constant
#define I1  0
#define I2  1
#define I3  2
#define I4  3
/*-------------------------------------------------------------
Function: integrate
Parameters:
    integralId - one of I1, I2, I3 or I4 to identify integral
                 to solve
    x1, x2 - limits of integration
Return Value
    The result of the definite integral
Description: Solves one of four defnite integrals and returns
             the result.
-------------------------------------------------------------*/
#define NUM_STEPS 20
double integrate(int integralId, double x1, double x2)
{ ... }
```

Solution to integrals: I1: 2.886, I2: 2066.24, I3: 98.992, I4: 41.61

8) The total mass of a variable density rod is given by

$$m = \int_0^L \rho(x) A_C(x) dx$$

where m = mass (kg), $\rho(x)$ = density (kg/m^3), $A_c(x)$ = cross-sectional area (m^2), x = distance along the rod (m) and L = the total length of the rod (m). Determine the mass in kilograms to the best possible accuracy. Assume that the density and cross-section area vary according to:

$$\rho(x) = 4000 - 70x$$

$$A_c(x) = 0.01 + 5 \times 10^{-5} x^2$$

(a) Apply the Trapezoidal Rule and develop the numerical equation for determining the mass of a rod for a given length L. Provide the numerical equation in the simplest form possible. Also show how the step h is calculate given a number of integration steps.

(b) Complete the following function that returns the mass of the rod in kg given its length. Use a symbolic constant to define the number of integration steps.
(Some sample valid values: L=2.0, mass=79.12; L=3.2, mass=126.51, L=5.6, mass=223.87; L=8.5, mass=351.09; L=10.0, M=422.92; note these values were obtained using the analytical solution, depending on the integration step, the results may not match exactly).
Consider using a separate function for computing the values of $f(x_i)$.

```
/*-------------------------------------------------------------
Function: findRodMass
Parameters:
    length - rod length
Return Value
    The mass of the rod.
Description: Applies the trapezoidal rule to solve for the
             mass of a rod whose density and cross-sectional
             area varies with length.
-------------------------------------------------------------*/
#define NUM_STEPS 20
double findRodMass(double length)
{ ... }
```

9) In electrical power engineering the RMS voltage of alternating current is an important measure of the electrical system. The alternating current is a sinusoidal voltage given by $A\sin(2\pi ft)$ with some given amplitude A and frequency f. Its RMS voltage is given by:

$$v_{rms} = \sqrt{\frac{1}{T}\int_0^T A^2 \sin^2(2\pi ft)dt}$$

Where the limits of integration over time, t (seconds), are defined by the period T (which is the inverse of the frequency f), and f is the voltage frequency (typically the frequency has the value of 60 Hz, that is, 60 cycles per second, in North American electrical grids).

(a) Apply the trapezoidal rule and develop a numerical equation for determining v_{rms} for a given amplitude and frequency. Make the equation as simple as possible. Also show how the step h is calculate given a number of integration steps.

(b) Use the results from (a) to complete the following function that returns the RMS voltage for a given amplitude and frequency. The analytical solution of the RMS voltage is quite simple, it is simply $A/\sqrt{2}$.

```
/*-------------------------------------------------------------
Function: rmsVoltage
Parameters:
    a - Signal amplitude A
    f - Signal frequency f
Return Value
    The RMS value of the AC voltage.
Description: Using the Trapezoidal rule, computes the
             RMS voltage of the AC voltage sin(2PI f t).
-------------------------------------------------------------*/
#define NUM_STEPS 20
double rmsVoltage(double a, double f)
{ ... }
```