ICS 103
Sample Quiz 4

Question 1 (14 POINTS):

Find the output of the following programs:

I)

```
#include <stdio.h>
int main (){
  int x = 6, y = 4, z;
  int *p1, *p2;
  p1 = &x;
  p2 = &y;
  z = x;
  x = y;
  y = z;
  printf("%d %d\n", *p1, *p2);
  *p1 = x+3;
  *p2 = ++x – y;
  printf("%d %d\n", x, y);
  p1 = p2;
  printf("%d %d\n", *p1, *p2);
  return 0;}
```

OUTPUT

```
4    6
8    2
2    2
```

II)

```
#include <stdio.h>

void change(int w, int x, int *y, int *z);

int main(void){
     int a = 3, b = 4;
     change(a * 2, b * 3, &a, &b);
     printf("%d %d\n", a * 2, b * 3);
     return 0;
}

void change(int w, int x, int *y, int *z){
     w = x;
     x = *y;
     *y = *z;
     printf("%d %d %d\n", w, x, *y);
     *z = w;
}
```

OUTPUT

```
12   3    4
8    36
```

See Next Page!

III)

```
#include <stdio.h>

int fun(int x);

int main(void) {
     printf("%d %d %d\n", fun(3), fun(2), fun(0));
     return 0;
}

int fun(int x){
     if (x == 3 || x == 4)
          return x - 1;
     else
          return x + fun(x+1) + fun(x+2);
}
```

OUTPUT

```
2     7     17
```

Question 2 (13 POINTS):

Write a **RECURSIVE** integer function **odd_count** that takes a positive integer input variable **n**, and then returns the count of odd digits in **n**. As an example, if **n** = 63827, **odd_count** function should return 2, this is because there are two odd digits in **n** = 63827 (these are 3 and 7).

Note:
Write an only recursive solution for **odd_count** function, do NOT use any loop. No credit will be given to a non-recursive solution. Don't write a main function.

Hint:
Recursively test the digits of **n** one by one, if the current digit is odd, then increment and repeat the same test for the remaining digits. In case the current digit is even, then only repeat the same test for the remaining digits without incrementing. You should stop when you finish testing all digits (**n** will be shrinking for every recursive call).

```
int odd_count(int n){                                   Header   3.0
  if (n == 0)                             Terminating Condition 2.0
      return 0;                                                 1.0
  else if (n % 10 % 2 != 0)                 Recursive Condition 3.0
      return 1 + odd_count(n / 10);                             2.0
  else
      return odd_count(n / 10);}                                2.0
```

See Next Page!

Question 3 (13 POINTS):

Write a function **evndiv** that receives a positive integer **n** and returns the **count** and the **sum** of the even divisors of **n** other than **n** itself. For example if the function receives 16, it returns 3 as the **count**, and 14 as the **sum**, this is because the even divisors of 16 are 2, 4 and 8. As another example, if **evndiv** receives 9, it should return 0 as the **count** and 0 as the **sum** because there are no even divisors for 9.

```
  void evndiv(int n, int *count, int *sum){                  Header   5.0

  int i;

  *sum = 0; *count = 0;                             Initializations   1.0

  for (i = 2; i <= n / 2; i += 2){                             Loop   3.0

      if (n % i == 0){                                    Condition   2.0

          *count = *count + 1;                                        1.0

          *sum = *sum + i;}                                           1.0

  }

}
```

Question 3 (9 POINTS): **1.5 for Each Output Value (Format is Important)**

Find the output of the following program:

```
#include<stdio.h>

double third(double j, double k){
  j++;
  k--;
  return j + k;}

void second(double *x, double *y, double z){
  z = third(*x,*y);
  printf("%0.2f\n",z);
  *y = 2.0 * z;
  *x = 2.0 * *y;}

void first(double *n, double *m, double h){
  *n = *n - *m;
  *m = h * *n;
  printf("%0.2f\t%0.2f\n",*n, *m);
  second(n,m,h);}

main(){
  double a = 6.0, b = 0.0, c = 4.0;
  first(&a,&b,c);
  printf("%0.2f\t%0.2f\t%0.2f\n",a,b,c);}
```

OUTPUT

```
6.00      24.00
30.00
120.00    60.00     4.00
```