CS 210 **Name_SOLUTION____**
Hour Exam 1 **September 22, 2016**

1. Show how many lines of print are printed by each of the following:

```
int i, j, k;
for(i=0;i<10;i++)
  for(j=0;j<20;j++)
    for(k=0;k<30;k++)
       printf("Hello Mom!\n");
```

Lines Printed = __6000_____

```
int i, j;
for(i=0;i < 10;i+=2)
   {printf("Hello Mom!\n");
    j = 16;
    while(j >= 0)
       {printf("I like Jello\n");
        j--;
       }
     printf("bananas are good.\n");
    }
```

Lines printed = __**95**_____

2. Write a line of C-code to implement the following equation. Take all variables to be doubles.

$$x = \frac{y + b/c - d^4}{13z + 12.5}$$

___`x = (y + b/c - pow(d, 4))/(13*z + 12.5);`___

3. If i = 3, j = 12, and k = 4, determine whether each of the following is TRUE or FALSE.

A) `(!((i+j)<20)&&(i==3))` _**FALSE**_ B) `((!(j==12)||!(k<7)))` **FALSE**___

4. What is printed by the following sequence.

```
int i = 3, j = 11;
double k;
k = j/i;
printf("%d, %2.3f", j/i,  k);
```

Printed results

___**3, 3.000**_____

_____

5. Give at least two advantages to using functions to modularize a program.

A) Saves memory if the function is called multiple times.
B) Provides a way to organize a program in smaller steps – divide and conquer.
C) Allows multiple programmers to implement pieces of code independently.

6. The statements below prompt the user to enter two integers called i and j. Write an *if block* to print the value of i and j only if the value of i is in the range ($10 \le i < 100$) *and* i is greater than j. If this is not the case your if block should print only the value of i.

```
int i, j;
printf("Enter an integer...");
scanf_s("%d", &i);
printf("Enter a second integer...");
scanf_s("%d", &j);
// Put your if block here
if(10 <= i && i < 100 && i > j)
   printf("%d %d\n", i, j);
else
  printf("%d\n", i);
```

7. Show what is printed by the following and fill in the memory map.

```
int Fun1(int a, int b);
int main()
{
   int a = 1, b = 2, c = 3;
   printf("%d %d %d\n", a, b, c);
   c = Fun1(a, b);
   printf("%d %d %d\n", a, b, c);
}
int Fun1(int x, int y)
{
   int a;
   a = x;
   x = y;
   y = a;
   printf("%d %d %d\n", x, y, a);
   return x;
}
```

Printed Results

1 2 3
2 1 1
1 2 2

Fun1	Main	Data
	a	1
	b	2
	c	~~3~~ 2
x		~~1~~ 2
y		~~2~~ 1
a		1

8. Write a program which prints the powers of 2 from 2^0 to 2^{16} on successive lines. *Do not use the pow function.*

```
int i, x = 1;
for(i=0;i<=16;i++)
{
      printf("%d\n", x);
      x *= 2;
}
```

9. Write a *function* which accepts two integer arguments name `max` and `min` and returns an `int`. Your function should input a number from the user and return that number if and only if it is greater than or equal to `min` AND less than or equal to `max`. Otherwise, it should return a 0. Name your function `MaxMin`.

```
int MaxMin(int max, int min)
{
    int n;
    printf("Enter an int ... ");
    scanf_s("%d", &n);
    if(n >= min && n <= max)
       return n;
    return 0;
}
```

CS 210 **Name SOLUTION**
Hour Exam 2 **October 18, 2016**

1. If an array is defined by the statement `int A[30];` , answer the following questions:

A) How many integers can be stored in the array? **30**

B) Write a statement to store the number 54 in the last element of the array.

```
A[29] = 54;
```

C) If the user write the statement below, what are the possible results? Explain.

```
A[34] = 22;
```

A[34] does not exist so this will likely cause a memory fault or possibly set another variable to the value of 22.

D) Write a short sequence of statements to exchange the values in element `A[12]` with the value in `A[11]`.

```
int tmp;
tmp = A[12];
A[12] = A[11];
A[11] = tmp;
```

2. C does not do bounds checking on array sizes. What does this mean and what are the implications for C programmers.

It means that when an array is defined with a size, the C-compiler does not check to see that all array operations stay within the size boundary. This may result in a program seeding access to a memory location which it does not own creating an error.

3. Show what is printed by the following program. Assume that the variable u is stored in memory at 7339360 and v is stored at 7339348.

```
#include<stdio.h>
int main()
  {int u = 3, v;
   int *ptrU, *ptrV;
   ptrU = &u;
   v = *ptrU;
   ptrV = &v;
   printf("u = %d v = %d\n", u, v);
   printf("%d, %d\n", ptrU, ptrV);
   printf("%d, %d\n", &u, &v);
   return 0;
  }
```

Printed Results

```
u = 3 v = 3
7339360, 7339348
7339360, 7339348
```

4. Mark each statement *A* to *D* below as true or false. The statement apply after the following code fragment runs.

```
int [] x = {9, 8, 7, 6};      //line 1
int [] y = {12, 15, 4, 3};    //line 2
x[2] = 4;                     //line 3
y[0] = x[3];                  //line 4
x[y[3]] = 12;                 //line 5
```

A) line 5 is illegal **FALSE**
B) x[1] = 8 **TRUE**
C) y[3] = 4 **FALSE**
D) y[0] = 9 **FALSE**

5. Show what is printed by the following program.

```
#include <stdio.h>
int MFun(int *, int);
int main()
 {int a = 5;
  int *aptr;
  aptr = &a;
  int b = 0, c = 0;
  c = MFun(aptr, b);
  printf("%d %d %d\n", a, b, c);
  return 0;
 }
//
int MFun(int *x, int y)
 {y = *x * *x;
  printf("%d %d\n", *x, y);
  return y;
 }
```

Printed results

5 25

5 0 25

6. Write a function which will return the average value of a one-dimensional int array passed as a parameter. Name your method FindAverage. You may pass the array size as a second parameter.

```
double FindAverage(int a[], int n)
  {int i, sum = 0;
   for(i=0;i<n;i++)
      sum += a[i];
   return (double)sum/n;
  }
```

7. The following statements creates an array and fills it with random numbers. Write a sequence to find and print the *array index* of the item which has the least absolute value in the array.

```
srand(23);
int a[203];
int i;
for(i=0;i<203;i++)
  a[i] = (rand() % 25) -12;
```

```
int min, minIndex;
minIndex = 0;
min = abs(a[minIndex]);
for(i=1;i<203;i++)
   {if(min > abs(a[i]))
       {min = abs(a[i]);
        minIndex = i;
       }
   }
printf("%d\n", minIndex);
```

8. The program below is a method which does a *select sort*. This method calls two other methods named *FindMin* and *Swap*. Answer the questions below about these three methods.

```
void SelectionSort(int d[], int n)
    {int i, j, minIndx;
     for(i=0;i<n-1;i++)
        {for(j=i;j<n;j++)
           {FindMin(d, n, i, &minIndx);
            Swap(&d[i], &d[minIndx]);
            }
        }
    }
void FindMin(int d[], int n, int start, int *minIndx)
   {int i;
    *minIndx = start;
    for(i=start;i<n;i++)
       {if(d[i] < d[*minIndx])
           *minIndx = i;
       }
   }
void Swap(int *a, int *b)
   {int tmp;
    tmp = *a;
    *a = *b;
    *b = tmp;
   }
```

A) What will be in the array x if we execute the following two statements?

```
int x[] = {2, 8, 10, 6, 4};
SelectionSort(x, 5);
```

x = {2, 4, 6, 8, 10};

B) Does the Sort program still work correctly if two entries of the array it is sorting have the same value? For example:

```
int y[] = {2, 8, 10, 8, 4};
SelectionSort(y, 5);
```

Explain why or why not?

Yes it does work correctly. If two values are the same they will appear adjacent to each other in the final sorted array.

C) Show how you could use the FindMin method to find and print the minimum of the array given by

```
int z[] = {2, 8, 10, 6, 4, 0, -6, 15};
```

```
int minIndx;
FindMin(z, 8, 0, &minIndx);
printf("%d\n", z[minIndx]);
```

D) What is in array x after the following sequence runs:

```
int x[] = {2, 8, 10, 6, 4};
Swap(*x[2], *x[0]);
```

x = {10, 8, 2, 6, 4);

CS 210 **Name_SOLUTION_____**
Hour Exam 2 **October 18, 2016**
Practical In Class

Write a program containing the main program below and the function to go with it. The function will accept an integer array and three integer arguments called *min*, *max*, and *SIZE*. The function will return the number of items in the array that are greater than min and less than max.

For example, if the array contains {0,1, 2, 3, 4, 5, 6, 7, 8, 9} and *min* = 4 and *max* = 7, your function should return 2 since only two numbers are between 4 and 7.

```
#include<stdio.h>
int FindHowMany(int a[], int min, int max, int SIZE);
int main()
{
    int a[] = {1, 4, -5, 9, 7, 2, 5, -4, 3, 12};
    int SIZE = 10;
    int min = 3, max = 9;
    int k;
    k = FindHowMany(a, min, max, SIZE);
    printf("Number between %d and %d = %d\n", min, max, k);
    return 0;
}
int FindHowMany(int a[],int min,int max,int SIZE)
{
    int i, cnt = 0;
    for(i=0;i<SIZE;i++)
    {
       if(a[i] > min && a[i] < max)
          cnt++;
    }
    return cnt;
}
```

CS 210 **Name**_____
Hour Exam 3 **November 17, 2016**

1. On the blank lines below show what each print statement in the following code prints.

```
char s1[80] = "Hello ";
char s2[] = "Alligators make ";
char s3[] = "good shoes.";
char s4[] = "abcdgoldfish";
strncat(s1, s2, 10);
printf("%s\n", s2);                    Alligators make
printf("%s\n", s1);                    Hello Alligators
strcpy(s1, s4);
printf("%s\n", s1);                    Abcdgoldfish
printf("%d\n", strlen(s1));            12
strncpy(s1, s2, 10);
printf("%s\n", s1);                    Alligatorssh
```

2. On the blank lines below show what each print statement in the following code prints.

```
char s1[] = "wXyz";
char s2[] = "Wxyz";
char s3[] = "abcd";
char s4[] = "abcd";
char s5[] = "1234";
printf("%d\n", s1, s2, strcmp(s1, s2));   1
printf("%d\n", s2, s1, strcmp(s2, s1));   -1
printf("%d\n", s1, s2, strcmp(s1, s2));   1
printf("%d\n", s3, s4, strcmp(s3, s4));   0
printf("%d\n", s5, s4, strcmp(s5, s4));   -1
printf("%d\n", s1, s5, strcmp(s1, s5));   1
```

3. Write a function which accepts a string, s and an integer, n as arguments. If n is 1 the function returns the number of digits in the string. If n is 2 it returns the number of spaces in the string. If n has any other value it returns -1.

```
int HowMany(char *s,int n)
   {int i, cnt = 0;
    if(n != 1 && n != 2)
       return -1;
    if(n == 1)
       {for(i=0;i<strlen(s);i++)
           if(isdigit(s[i]))
              cnt++;
        return cnt;
       }
    for(i=0;i<strlen(s);i++)
       if(s[i] == ' ')
         cnt++;
    return cnt;
}
```

4. A struct has been defined as:

```
typedef struct
  {double x;
   double y;
  }myType_t;
```

A pointer to the struct is passed to a function. The first line of the function definition is:

```
void MyFun(myType_t *mp)
```

Show how the function can set the value of x in the parameter to 5.3.

```
mp -> x = 5.3;   or   (*mp).x = 5.3;
```

5. Write a function called *Max* which will return the maximum value stored in a two dimensional array. The array is of type *double* and has 15 rows and 32 columns. Your function should accept the array along with two integers which give its row and column dimension.

```
double Max(double a[][32], int row, int col)
  {int r, c;
   double maximum = a[0][0];
   for(r=0;r<row;r++)
     {for(c=0;c<col;c++)
        if(maximum < a[r][c])
            maximum = a[r][c];
     }
   return maximum;
  }
```

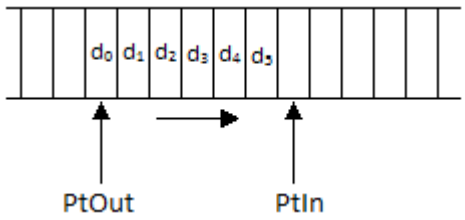
6. Write a function which receives a string argument and returns the number of upper case characters in the string. Name your function *CountUpper*.

```
int CountUpper(char *s)
   {int i, cnt = 0;
    for(i=0;i<strlen(s);i++)
       {if('A' <= s[i] && 'Z' >= s[i])
            cnt++;
       }
    return cnt;
   }
```

7. A queue has two pointers PtOut and PtIn. When something is placed into the queue it goes at the location PtIn and PtIn is incremented by 1. When something is taken out of the queue it is removed from the location pointed to by PtOut after which PtOut is incremented by 1. The user sees this queue as an abstract data type and has access only to the two pointers. Answer the following questions:

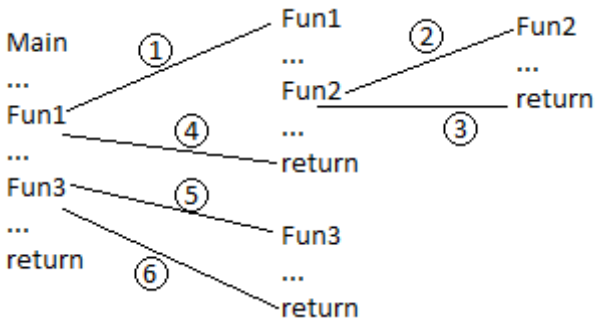
A) How can we determine if the queue is empty?
If PtOut = PtIn the queue is empty.

B) How can we determine how many data items are in the queue?
The difference PtIn – PtOut is the number of data items in the queue.



8. A main program calls Fun1 and Fun3. Fun1 also calls Fun2. If return addresses are saved on the stack and each return address takes one location, what is the maximum number of return addresses on the stack at any one time? Explain.

There will be two return addresses on the stack when main calls Fun1 and Fun1 calls Fun2.



9. This problem has three parts:

A) Prompt the user to enter an integer. Use scanf_s to put the user's integer into a variable called *n*. Check to see that *n* is in the range $5 \leq n \leq 30$. If *n* is outside this range set its value to 20.

```
int status, n;
printf("Enter a positive integer... ");
status = scanf_s("%d", &n);
if(!(n >= 5 && n <= 30)
   n = 20;
```

B) Create a dynamic array of chars of size *n*. Name this array myChars[]

```
int *myChars;
myChars = (char *)calloc(n, sizeof(char));
```

C) Write a loop to fill myChars with random upper case letters.

```
int i;
for(i=0;i<n;i++)
   myChars[i] = (char)(rand() % 26 + (int)'A');
```

CS 210 **Name_SOLUTION_____**
Hour Exam 3 **November 17, 2016**
Practical In Class
SOLUTION

Write a string function called *indexOf* which accepts a string, a starting position, and a character as arguments. It searches the string beginning with the starting position for the character and returns the index of the character. For example, if the string is "Hello Mom" and the function is called with

```
indexOf("Hello Mom", 5, 'o')
```

it would return 7.

If the starting position is greater than the length of the string or if the character is not found, your function should return -1.

Add comments to the top of your source code to indicate:

```
//Your name
//November 17, 2016
//Exam 3 Practical
```

Turn in a printed copy of your source code.

```
int IndexOf(char *s,int start,char c)
{
    int len = strlen(s);
    if(start > len)
       return -1;
    int i = start;
    while(i < len && s[i] != c)
       i++;
    if(i == len)
       return -1;
    return i;
}
```

CS 210 **Name_SOLUTION_____**
Hour Exam 3 **Nov. 17, 2016**
Practical In Class
SOLUTION

Write a string function called *InsertChar* which accepts a string, a position number, and a character as arguments. It inserts the character into the string at the position number. For example, if the string is "abcde" and the function is called with

```
int status = InsertChar(s, 3, 'x');
```

the string *s* would be modified to be "abcxde".

If the position number is greater than the string length, your function should return a status of -1. Otherwise, it should return a status of +1 to indicate success.

You may assume that the original string is big enough to accommodate the additional character without error.

Add comments to the top of your source code to indicate:

```
//Your name
//November 17, 2016
//Exam 3 Practical
```

Turn in a printed copy of your source code.

```
int InsertChar(char *s,int pos,char c)
   {int len = strlen(s);
    if(pos >= len)
       return -1;
    int i;
    for(i=len;i>=pos;i--)
       s[i+1] = s[i];
    s[pos] = c;
    return 1;
   }
```