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;
  }
```

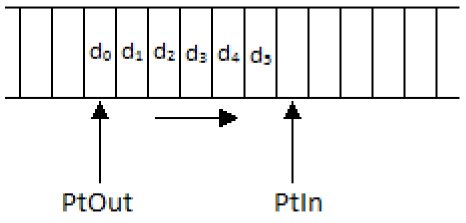
6. Write a function which receives a string argument and returns the number of upper case characters in the string. Name your function *CountUpper*.

```
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:

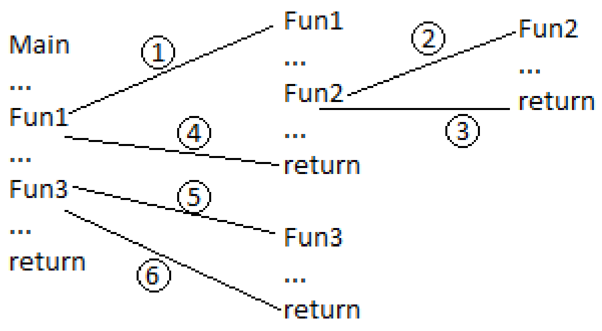
A) How can we determine if the queue is empty?
If PtOut = PtIn the queue is empty.

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 $5 \le n \le 30$. 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');
```