1. (8 points) **Input/Output**. Provide the *exact output* of the program shown below.

```
#include <stdio.h>
#include <string.h>

int main(int argc, char* argv[])
{
        char a[] = "abcdef";
        char b[] = "uvwxyz";
        char s[10];
        int i;

        for (i = 0; i < 5; i++) {
                if (i % 2 == 0)
                        strcpy(s+i, a+i);
                else
                        strcpy(s+i, b+i);

                printf("%s\n",s);
        }

        return 0;
}
```

Standard output:

abcdef

avwxyz

avcdef

avcxyz

avcxef

2. (8 points) **Input/Output**. Provide the *exact output* of the program shown below.

```
#include <stdio.h>

char d(int d)
{
        if (0 <= d && d <=9)
                return d + '0';
        else
                return d - 10 + 'A';
}

void c(int x, int b, char s[])
{
        int i;

        for (i = 0; x > 0; i++) {
                s[i] = d(x % b);
                x = x / b;
        }

        s[i] = '\0';
}

int main() {
        char s[100];

        c(11,2,s);
        printf("%s\n",s);

        c(61,16,s);
        printf("%s\n",s);

        c(12345,10,s);
        printf("%s\n",s);

        return 0;
}
```

Standard output:

1 1 0 1

D 3

5 4 3 2 1

3. (8 points) **Statement coverage**. In `main`, write integer constants in the empty boxes to achieve 100% statement coverage of function f.

```
f(int *x, int *y, int *z)
{
        if (*x > *y) {
                *x = *x;
                *x = *y;
                *y = *x;
        }

        if (*y > *z) {
                *x = *y;
                *y = *z;
                *z = *x;
        }

        if (*x > *y) {
                *x = *x;
                *x = *y;
                *y = *x;
        }
}

int main(int argc, char* argv[])
{
        int a = [ 5 ] ;

        int b = [ 3 ] ;

        int c = [ 1 ] ;

        f(&a, &b, &c);

        return 0;
}
```

4. (8 marks) **Input/Output**.

(a) Provide the *exact output* of the Python code shown below:

```
X = range(3,7)
print X[0]
print X[2:5]
print X[:3]
```

Standard output:

3

[5, 6]

[3, 4, 5]

(b) Provide the *exact output* of the Python code shown below:

```
X = range(3)
Y = [X,range(4)]
print Y

X[2] = 9
print Y

print Y[1][3]
```

Standard output:

[[0, 1, 2], [0, 1, 2, 3]]

[[0, 1, 9], [0, 1, 2, 3]]

3

5. (8 points) **Input/Output**.

(a) Provide the *exact output* of the program shown below.

```
def f(x):
        if x == [ ]:
                return [ ]
        return f(x[1]) + [x[0]] + f(x[2])

print f([ ])
print f([ 1, [], [] ])
print f([ 1, [2,[],[]], [3,[],[]] ])
print f([ 1, [2, [4,[],[]] ,[]], [3,[],[]] ])
```

Standard output:

[ ]

[1]

[2, 1, 3]

[4, 2, 1, 3]

(b) Provide a better name for function `f`. Choose a name which describes the computation carried out by `f`, as commonly understood by computer programmers.

inorder_traversal

6. (10 points) **Implementation to specification**. Provide a correct implementation for the `left_shifts` function specified below. Be sure that your code is as short and simple as possible.

```
# Purpose:
#       Return a list consisting of all of the left circular shifts of L
# Preconditions:
#       L is a list of integers
# Examples:
#       if L is [ ] then left_shifts(L) returns      [ ]
#       if L is [1] then left_shifts(L) returns      [ [1] ]
#       if L is [1,2] then left_shifts(L) returns    [ [1,2], [2,1] ]
#       if L is [1,2,3] then left_shifts(L) returns
#            [ [1,2,3], [2,3,1], [3,1,2] ]
#       if L is [1,2,3,4] then left_shifts(L) returns
#            [ [1, 2, 3, 4], [2, 3, 4, 1], [3, 4, 1, 2], [4, 1, 2, 3] ]
def left_shifts(L):

    L0 = [ ]

    for i in range(0,len(L)):
        L0.append( L[i:] + L[:i] )

    return L0
```

BONUS (5 points) **Implementation to specification**. Provide a correct C implementation for the `reverse` function specified below. Be sure that your code is as short and simple as possible.

Note: you must perform the reversal in place, that is, you may *not* define a character array as a local variable in `reverse`.

```
/*purpose
*       reverse the characters in s
* preconditions
*       s is a legal C string
* examples
*       s is a legal C string
*/
void reverse(char s[])
{

   int i,j;

   i = 0; // start of s
   j = strlen(s) - 1;
   while (i < j) {
            // swap s[i] and s[j]
            int c = s[i];
            s[i] = s[j];
            s[j] = c;

            // move i forward and j backward
            i++;
            j--;
   }
```

_____

_____

_____

_____

_____

_____

_____

_____

_____

```
}
```