1. (4 points) **Input/Output**. Provide the *exact output* of the program shown below.

```
#include <stdio.h>

int main()
{
         int i,j;

         for (i = 0; i < 5; i++) {
                  for (j = 0; j < 5; j++) {
                           printf("%d ",j);
                           if (i <= j)
                                    break;
                  }
                  printf("\n");
         }

         return 0;
}
```

Standard output:

0

0 1

0 1 2

0 1 2 3

0 1 2 3 4

2. (6 points) **Input/Output**. Provide the *exact output* of the program shown below.

```
#include <stdio.h>

int find(int x,int a[],int n)
{
          int i;

          for (i = 0; i < n; i++) {
                    if (x == a[i])
                              return i;
          }

          return -1;
}
int main(int argc, char* argv[])
{
          int b[] = {1,3,5,7,9};

          printf("%d\n",find(7,b,5));

          printf("%d\n",find(6,b,5));

          printf("%d\n",find(5,b+1,3));

          printf("%d\n",find(6,b+2,2));

          printf("%d\n",find(7,b+3,1));

          return 0;
}
```

Standard output:

```
3
-1
1
-1
0
```

3. (10 points) **Input/Output**. Provide the *exact output* of the program shown below.

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

/* convert s to integer; version 2 */
int atoi(char s[])
{
        int i, n, sign;

        for (i = 0; isspace(s[i]); i++)
                ; /* intentionally empty */
        printf("%d ",i);
        if (s[i] == '-')
                sign = -1;
        else
                sign = 1;
        if (s[i] == '+' || s[i] == '-')
                i++;
        printf("%d ",i);
        for (n = 0; isdigit(s[i]); i++) {
                n = 10 * n + (s[i] - '0');
                printf("%d ",n);
        }
        printf("\n");
        return sign * n;
}

int main(int argc, char *argv[])
{
        atoi("1");
        atoi(" +12 ");
        atoi("-12");
        atoi("  123x");
        atoi(" -12x3");

        return 0;
}
```

Standard output:

```
0 0 1
1 2 1 12
0 1 1 12
2 2 1 12 123
1 2 1 12
```

4. (8 points) **Legal strings and calls**. Answer the 10 true/false questions below. In each case, base your answer on the variable values at the point in the code at which the true/false question appears.

(a)

```
int main(int args,char *argv[])
{
        char s[10];
        char *t;

        s[0] = 'x';
        s[1] = 'y';
        // s is a legal C string: T or F: [F]
        s[0] = '\0';
        // s is a legal C string: T or F: [T]
        s[0] = 'a';
        s[9] = '\0';
        // s is a legal C string: T or F: [T]
        t = s+2;
        // t is a legal C string: T or F: [T]
        s[3] = '\0';
        // t is a legal C string: T or F: [T]

        return 0;
}
```

(b)

```
/* Purpose
 * if x is present in a[0..n-1]
 *      return i such that a[i] == x
 * else
 *      return -1
 * Precondition
 *      n >= 0
 *      a[0..n-1] legally addressable
*/
int find(int x, int a[], int n);

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

        find(1,b,5);     // this is a legal call: T or F [T]

        find(2,b,2);     // this is a legal call: T or F [T]

        find(3,b+2,3);   // this is a legal call: T or F [T]

        find(4,b+4,2);   // this is a legal call: T or F [F]

        find(5,b+4,1);   // this is a legal call: T or F [T]

        return 0;
}
```

5. (12 points) **Find-the-failure/fix-the-failure**. Study the specification and *incorrect* implementation of the `trim_left_right` function below.

(a) Given `char s[] = " abc ";`, provide the output of `printf("%s",s);` just after the call `trim_left_right(s)`, using the *incorrect* code shown below:

bc

(b) Provide a correct implementation of `trim_left_right` by modifying the code. Mark your corrections clearly on the code. Make as few changes as possible.

```
/* Purpose:
 *        remove leading and trailing blanks and tabs
 * Preconditions:
 *        s is a legal C string
 * Examples:
 *        if s is "abc " then
 *               after calling trim_left_right(s), s is "abc"
 *        if s is "\t \tabc " then
 *               after calling trim_left_right(s), s is "abc"
 */
void trim_left_right(char* s)
{
        int i,start,end;

        for (start = 0; s[start] == ' ' || s[start] == '\t'; start++)
               ; // intentionally empty

        end = start;
        for (i = start; s[i] != '\0'; i++) {
               s[i-start~~-1~~] = s[i];
               if (s[i] != ' ' && s[i] != '\t') {
                      end = i;
               }
        }
        ~~s[end-start+1] = '\0';~~
                                  if (s[start] != '\0')
                                      s[end-start+1] = '\0';
                                  else
                                      s[0] = '\0';
}
```

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

```
/* Purpose:
        if a[0..n-1] contains two consecutive values which are equal
                return the index of the first element of the leftmost pair
        else
                return -1
Preconditions:
        a[0..n-1] is legally addressable
Examples:
        Given: int a0[] = {1,2,2,1,1};
        two_equal(a0,5) should return 1
        two_equal(a0+2,3) should return 1
        two_equal(a0+3,2) should return 0

        two_equal(a0,0) should return -1
        two_equal(a0,1) should return -1
        two_equal(a0,2) should return -1
*/
int two_equal(int a[],int n)
{

    int i;

    if (n < 2)
            return -1;

    for (i = 0; i < n-1; i++) {
            if (a[i] == a[i+1]) {
                    return i;
            }
    }
    return -1;








}
```

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

```
/*
Purpose
        Remove all occurrences of c from s
Preconditions
        s is a legal C string
Example
        given: char s[] = "xyxxa";
        after calling squeeze(s,'x')
                strcmp(s,"y") will return 0
*/
void squeeze(char s[],char c)
{


int i,j;

for (i = j = 0; s[i] != ’\0’; i++)
        if (s[i] != c)
                s[j++] = s[i];
s[j] = ’\0’;

}
```