Engineering 12 - Spring, 1999

Final Exam

Name:____

Section:	**Chelemer**	**Cugini**	**Gottfried**
(Circle one)	**Jacobs - 9 am**	**Jacobs - 1 pm**	**Lin**
	Needy	**Vasko**	**Vipperman**

This exam consists of Six (6) questions. Some questions count more than others.

Write all answers in this exam booklet.

Question 1: (12 pts)

Question 2: (15 pts)

Question 3: (16 pts)

Question 4: (16 pts)

Question 5: (16 pts)

Question 6: (25 pts)

____

Total:

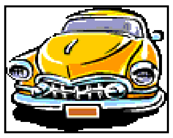
1. (12 points) A complete C program is given below.

```
#include <stdio.h>

main()
{
     int u = 3;
     int v = 5;
     int t = 1;
     int *pv = &u;
     int *pu = &v;
     int *pt = &t;

     printf("u = %d  *pu = %d\n",   u, *pu);
     printf("v = %d  *pv = %d\n\n", v, *pv);

     pt = pu;
     pu = pv;
     pv = &v;
     *pv = *pt;

     printf("u = %d  *pu = %d\n",   u, *pu);
     printf("v = %d  *pv = %d\n\n", v, *pv);
}
```

(a) What output is generated by the first two `printf` statements?

(b) What output is generated by the last two `printf` statements?

2. (15 points) Consider the image shown at the right. It consists of a grid of 64 × 64 pixels, each of which is black, white, or some shade of gray. Suppose this image is represented within a C program as a two-dimensional array of integers, declared as



```
int image[64][64];
           ↑   ↑
           |   column
          row
```

The integers in the array take on values between 0 and 255, where 0 represents black and 255 represents white. Other values represent shades of gray (for example, 1 is almost black, 127 is medium gray, 254 is almost white, etc.)

Shown on the next page are three different functions, each of which accepts the image array as input, and modifies the image in some fashion by assigning different values to the array elements. Each function performs one of the following image modifications:

(a) Turns the entire image black.

(b) Turns the entire image white.

(c) Inverts the image, turning white pixels black, black pixels white, and setting all gray pixels to their opposite gray level.

(d) Produces a mirror image, reversing left and right.

(e) Produces a mirror image, reversing top and bottom.

(f) Rotates the image 90 degrees counter-clockwise.

Note that these functions will not be applied consecutively to the image, but should each be considered separately.

Specify what each function does by selecting the correct letter from the above list.

Write the letter in the space labeled *purpose* at the end of each function.

```
void ip1(int image[][64])
{
     int i, j;

     for (i=0; i<64; i++)
         for (j=0; j<64; j++)
             image[i][j] = 0;

     return;
}
```

Purpose: ________

```
void ip2(int image[][64])
{
     int i, j;

     for (i=0; i<64; i++)
         for (j=0; j<64; j++)
             image[i][j] = 255 - image[i][j];

     return;
}
```

Purpose: ________

```
void ip3(int image[][64])
{
     int i, j, temp;

     for (i=0; i<64; i++)
        for (j=0; j<32; j++)   {
            temp = image[i][j];
            image[i][j] = image[i][63-j];
            image[i][63-j] = temp;

       }
     return;
}
```

Purpose: ________

3. (16 points) A game is played in which a single die is tossed. If the result is 4 or 6, the player wins. If the result is 1 or 3, the player loses. If the result is 2 or 5, the player continues to roll the die until a win or a loss occurs.

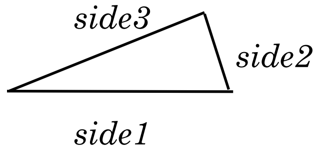
A complete C function called `game` is given below, simulating the outcome of rolling the die one time. Rewrite the function so that it uses `switch` and `case` statements rather than nested `if-else` statements.

```
void game(int roll)
{
     if ((roll == 1) || (roll == 3))
        printf("%d - You LOSE\n", roll);
     else if ((roll == 4) || (roll == 6))
        printf("%d - You WIN\n", roll);
     else
        printf("%d - Roll Again\n", roll);

     return;
}
```

4. (16 points) The following program will read in the lengths of the sides of a triangle (*side1*, *side2*, and *side3*) and then compute its area. Recall the following formulas:

$$perimeter = side1 + side2 + side3$$

$$s = perimeter\ /\ 2$$



$$area = \sqrt{s(s - side1)(s - side2)(s - side3)}$$

The program consists of four functions, called `main`, `input`, `compute`, and `output`. If you examine the program closely, you will note that it is well commented.

Some of the code is missing. Fill in the missing code in the spaces provided.

```
    /*  Program to calculate the area of a triangle  */

    #include<stdio.h>
    #include<math.h>


    void input(float side[]);                   /* Input Function Prototype   */

(a) _____________________________________       /* Compute Function Prototype */

(b) _____________________________________       /* Output Function Prototype  */


    main()
    {
            float side[3], area;

(c)         __________________________          /* Input Function Access   */

            area = compute(side);               /* Compute Function Access */

(d)         __________________________          /* Output Function Access  */
    }


(e) ______________________________________      /*Input Function Definition */
    {
            printf("\n Enter the length of the first side of the triangle: ");
            scanf("%f", &side[0]);
            printf("\n Enter the length of the second side of the triangle: ");
            scanf("%f", &side[1]);
            printf("\n Enter the length of the third side of the triangle: ");
            scanf("%f", &side[2]);

(f)         ___________________________
    }


(g) _______________________________________     /* Compute Function Definition */
    {
            float perimeter, s, area;

            perimeter = side[0] + side[1] + side[2];
            s = perimeter / 2;
            area = sqrt(s * (s - side[0]) * (s - side[1]) * (s - side[2]));

(h)         ___________________________
    }


    void output (float area)                    /* Output Function Definition */
    {
            printf("\nArea = %.2f", area);

            return;
    }
```

5. (16 points) The C program shown below is supposed to read several Fahrenheit-scale temperature values from the keyboard, convert each temperature to Celsius, and write the converted Celsius temperatures to a data file called "`ctemps.dat`." The process is supposed to continue until the sentinel 9999 is entered from the keyboard.

The program contains ***four errors***. Circle each line that contains an error, label it A, B, C or D, and write the corrected version of each line (with its appropriate label) at the bottom of the page.

```
/* Fahrenheit to Celsius temperature conversion */

#include <stdio.h>

main()
{
      float F, C;
      FILE fpt;

      fpt = fopen("ctemps.dat", "r");

      /* enter first temperature */
      printf("\nTemperature, in degrees F: ");
      scanf("%f", &F);

      while (F != 9999)   {
                C = (5.0 / 9.0) * (F - 32.0);
                fprintf("Temperature, in degrees C: %f\n", C);

                /* enter next temperature */
                printf("\nTemperature, in degrees F: ");
                fscanf("%f", &F);
      }

      printf("\nBye, Have a Nice Exam!");
      fclose(fpt);
}
```

6. (25 points) The attached program has been suggested as a solution to the last homework assignment (solve an algebraic equation using the method of bisection). The lines have been numbered for your convenience.

   (a) Complete line 21

   (b) Complete line 27

   (c) Complete lines 67 and 76 (they are identical)

   (d) Complete lines 68 and 77 (they are identical)

   (e) Complete line 81 (or answer “leave as is”).

```c
/* root solving using the method of bisection */

#include <stdio.h>
#include <math.h>

#define PI 3.1415927

float evaluate(float x);
float input(void);
int solution(float interval, float *xl, float *xr, float *xmid, float *fmid);
void output(int n, float *pxl, float *pxr, float *px, float *pf);





main()
{
       int n;
       float interval, xl = 0, xr = 1, x, f;
       float *pxl = ____, *pxr = ____, *px = ____, *pf = ____;

       interval = input();

       n = solution(interval, pxl, pxr, px, pf);

       output(_____, _____, _____, _____, _____);
}





float evaluate(float x)
{
       float f;

       f = 1.0 + exp(-x / (2 * PI)) * sin(x) - 1.25;

       return(f);
}





float input(void)
{
       float interval;

       printf("Please enter a value for the final interval: ");
       scanf("%f", &interval);

       return(interval);
}
```

```c
int solution(float interval, float *xl, float *xr, float *xmid, float *fmid)
{
       float fl, fr;
       int n = 0;

       *xmid = ________________________;
       *fmid = ________________________;
       while ((*xr - *xl) > interval)   {
             fl = evaluate(*xl);
             fr = evaluate(*xr);
             if (fl * (*fmid) > 0)   /* keep right interval */
                   *xl = *xmid;
             else                    /* keep left interval */
                   *xr = *xmid;
             *xmid = ________________________;
             *fmid = ________________________;
             n++;
       }

       return(_____);
}




void output(int n, float *pxl, float *pxr, float *px, float *pf)
{
       printf("Final answer:  x = %f   f(x) = %f\n", *px, *pf);
       printf("Final interval: xl = %f   xr = %f\n", *pxl, *pxr);
       printf("No. of iterations: %d", n);

       return;
}

```