2. It is your responsibility to write legibly. Unclear answer will be marked as wrong.
3. It is crucial to show all your work on the provided paper. Partial credits may be granted based on your work if it is not fully correct.
4. The bonus will be added to the total score, but not exceeding 100.
5. Manage your time, as the problems are not sorted by their difficulties.

<table>
<thead>
<tr>
<th>Problem 1</th>
<th>Problem 2</th>
<th>Problem 3</th>
<th>Problem 4</th>
<th>Problem 5</th>
<th>Problem 6</th>
<th>Problem 7</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>(15 pts)</td>
<td>(15 pts)</td>
<td>(12 pts)</td>
<td>(5 pts)</td>
<td>(12 pts)</td>
<td>(20 pts)</td>
<td>(21 pts)</td>
<td>(100 pts)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bonus: 5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bonus: 5</td>
<td></td>
</tr>
</tbody>
</table>
1. (15 points) 2-dimensional array

Complete the following program such that it will (1) print out the value of each element in the array, (2) count and print out the number of non-zero elements in the array. Also give the exact output of your completed program.

```c
int main()
{
    int a[3][3] = {{2,3}, {0}, {1, 2, 3}, {0, 5}};
    int counter;

    Exact output of your completed program:

    __ __ __ __  __ __ __ __ __ __ __ __  __ __ __ __ __ __ __ __ __
    __ __ __ __  __ __ __ __ __ __ __ __  __ __ __ __ __ __ __ __ __
    __ __ __ __  __ __ __ __ __ __ __ __  __ __ __ __ __ __ __ __ __
    __ __ __ __  __ __ __ __ __ __ __ __  __ __ __ __ __ __ __ __ __
    __ __ __ __  __ __ __ __ __ __ __ __  __ __ __ __ __ __ __ __ __
    __ __ __ __  __ __ __ __ __ __ __ __  __ __ __ __ __ __ __ __ __
    __ __ __ __  __ __ __ __ __ __ __ __  __ __ __ __ __ __ __ __ __
    __ __ __ __  __ __ __ __ __ __ __ __  __ __ __ __ __ __ __ __ __
```
2. (15 points)  loop, break, and continue

Give the exact output of the following program and rewrite by not using the `break` and `continue` statements.

```c
#include <stdio.h>
int main(void)
{ int x, y;
  for (x=1; x<10; x++)
  { if (x == 5)
      break;
      printf("%2d", x);
  }
  printf("\nx =%2d\n",x);
  for (y=1;y<10; y++)
  { if (y == 5)
      continue;
      printf("%2d", y);
  }
  printf("\ny =%2d\n",y);
  return 0;
}
```

Output:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

Your program without `break` and `continue` statements:
3. (12 points, 2 points per printf()) pointers

Give the exact output of the following program. Write your answer on the provided area, with one letter per space. Use ‘–1’ to indicate elements that are not assigned values.

```c
#include <stdio.h>
int main(void)
{
    int arr[4] = {10, 20, 30, 40};
    int *ptr = arr;
    printf("%d\n", *ptr);
    ptr++;
    printf("%d\n", *ptr);
    printf("%d\n", *(ptr + 2));
    printf("%d\n", *ptr++);
    printf("%d\n", *ptr);
    return 0;
}
```

Output:

```
__ __ __ __  __ __ __ __ __ __ __
__ __ __ __  __ __ __ __ __ __ __
__ __ __ __  __ __ __ __ __ __ __
__ __ __ __  __ __ __ __ __ __ __
__ __ __ __  __ __ __ __ __ __ __
__ __ __ __  __ __ __ __ __ __ __
```

4. (5 points) static variable

Give the exact output of the following program. Write your answer on the provided area, with one letter per space. Use ‘–1’ to indicate elements that are not assigned values.

```c
#include <stdio.h>
void increase(int x)
{
    static int i = 0;
    x = i + x;
    i++;  
    printf("x=%d, i=%d\n", x, i);
}
int main(void)
{
    int i = 0;
    for (; i < 4; ++i)
    {
        increase (i);
        return 0;
    }
}
```

Output:

```
__ __ __ __  __ __ __ __ __ __ __ __ __ __
__ __ __ __  __ __ __ __ __ __ __ __ __ __
__ __ __ __  __ __ __ __ __ __ __ __ __ __
__ __ __ __  __ __ __ __ __ __ __ __ __ __
__ __ __ __  __ __ __ __ __ __ __ __ __ __
__ __ __ __  __ __ __ __ __ __ __ __ __ __
__ __ __ __  __ __ __ __ __ __ __ __ __ __
```

Page 4 of 7
5. (12 points) recursion

Consider the function defined as follows:
\[ f(0) = 1 \]
\[ f(1) = 3 \]
\[ f(n) = 2f(n-1) - f(n-2) \quad \text{if } n > 1 \]

1) (4 points) Compute (by hand) the followings
\[ f(3) = \quad f(4) = \quad f(5) = \quad f(6) = \]

2) (8 points) Using recursive function to implement this function.
6. (20 points) array

1) (18 points) Give the exact output of the following program. Write your answer on the provided area, with one letter (including space and punctuations) per space. There is no empty space in the “ “ of the printf() command. Use “–1” to indicate elements that are not assigned values.

```c
#include <stdio.h>
int main(void)
{
    int i=0, done=0, temp;
    int a[] = {5, 3, 8, 2, 7};
    while (!done)
    {
        printf("Start:");
        done = 1;
        for (i = 0; i < sizeof(a)/sizeof(a[0])-1; i++)
            if ( a[i] > a[i+1])
                { temp = a[i];
                  a[i] = a[i+1];
                  a[i+1] = temp;
                  done = 0;
                  printf("%d," , a[i]);
                }
        printf("end.\n");
    }
    printf("Size_of_a=%d\n", sizeof(a)/sizeof(a[0]));
    i = 0;
    do
    {
        printf("%d," , a[i]);
        i++;
    } while (i < sizeof(a)/sizeof(a[0]) - 1);
    return 0;
}
```

Output:

```
  __ __ __ __  __ __ __ __ __ __ __ __ __ __ __ __
  __ __ __ __  __ __ __ __ __ __ __ __ __ __
   __ __
  __ __ __ __  __ __ __ __ __ __ __ __ __ __ __ __
  __ __ __ __  __ __ __ __ __ __ __ __ __ __
   __ __
  __ __ __ __  __ __ __ __ __ __ __ __ __ __ __ __
  __ __ __ __  __ __ __ __ __ __ __ __ __ __
   __ __
  __ __ __ __  __ __ __ __ __ __ __ __ __ __ __ __
  __ __ __ __  __ __ __ __ __ __ __ __ __ __ __ __
   __ __
```

2) (2 points) What does the while loop do? Explain in a couple of sentences.
7. (26 points with 5 for bonus, 3 points each for 1) and 2), 5 points each for the rest)

```c
int Binary_Search(int a[], int n, int key)
{
    int left = 0, right = n-1, check;
    while (left <= right )
    {
        check = (left + right)/2;
        if (a[check] == key)
            return check;
        if (a[check] < key)
            left = check + 1;
        if (a[check] > key)
            right = check - 1;
    }
    return -1;
}
```

Answer the following questions on this binary search function from the lecture notes. Use the back of the previous page and write down the question number if you need more space.

1) what does this function return?

2) what assumption do we have to make on the array?

3) can we initialize variable “right = n” in line 2? why?

4) can we replace the condition “left <= right” by “left < right” in line 3? why?

5) can we replace line 4 by “check = (left + right)/3” ? why?

6) can we replace line 8 by “left = check” ? why?