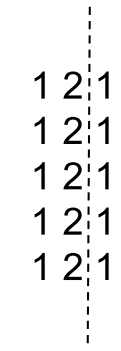
Chapter 5: Nested Loops in C – Exercise Answers

Question 1: Write C programs to display the following patterns using nested loop construct

a) Pattern: 1 2 3 (repeated 4 times)

```
#include <stdio.h>
int main()
{
    int i, j;
    for (i = 1; i <= 4; i++)
    {
        for (j = 1; j <= 3; j++)
        {
            printf("%d ", j);
        }
        printf("\n");
    }
    return 0;
}
```

b) Pattern: 1 2 1 (repeated 5 times)




```
#include <stdio.h>
int main()
{
    int i, j;
    for (i = 1; i <= 5; i++)
    {
        // First part: print "1 2"
        for (j = 1; j <= 2; j++)
        {
```

```
            printf("%d ", j);
        }

        // Second part: print "1"
        for (j = 1; j <= 1; j++)
        {
            printf("1");
        }

        printf("\n");
    }
    return 0;
}
```

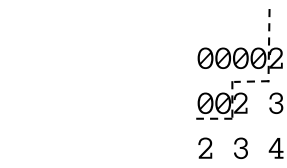
c) Pattern: 4 3 2 1 (repeated 5 times)

Note: i-- decrements the value of i by 1.

```
#include <stdio.h>
int main()
{
    int i, j;
    for (i = 1; i <= 5; i++)
    {
        for (j = 4; j >= 1; j--)
        {
            printf("%d ", j);
        }
        printf("\n");
    }
    return 0;
}
```

d) Pattern: Right-aligned number triangle

```
    2
  2 3 4
2 3 4 5 6
```



0 => spaces (just for counting purposes)

spaces

i	OUTPUT	j
1	0000	4
2	00	2
3		0

AP -> 4,2,0

d= 2-4 = -2

$a_i = a + (i-1)d$

$= 4 + (i-1)(-2)$

$= 6 - 2*i$

numbers

i	OUTPUT	j
1	2	1
2	2 3 4	3
3	2 3 4 5 6	7

AP -> 1, 3, 7

d = 3-1 = 2

$a_i = a + (i-1)(d)$

$= 1 + (i-1)2$

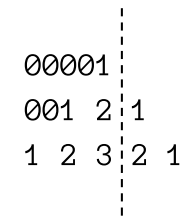
$= 1 + 2*i - 2$

$= -1 + 2*i$

```
#include <stdio.h>
int main()
{
    int i, j, k;
    for (i = 1; i <= 3; i++)
    {
        // Print spaces
        for (j = 1; j <= (6-2*i); j++)
        {
            printf(" ");
        }
        // Print numbers
        for (k = 1; k <= (-1 + 2*i); k++)
        {
            printf("%d ", k+1);
        }
        printf("\n");
    }
    return 0;
}
```

e) Pattern: Diamond number pattern

```
    1
  1 2 1
1 2 3 2 1
```



spaces are taken as 0's (so its easier to count for calculating)

i	**spaces**	**j**
1	0000	4
2	00	2
3		0

AP -> 4, 2, 0
d = 2-4 = -2
a_i = a + (i-1)(d)
= 4 + (i-1)(-2)
= 4 - 2*i + 2
= 6 - 2*i

ascending numbers

i	**OUTPUT**	**j**
1	1	1
2	1 2	2
3	1 2 3	3

AP -> 1, 2, 3
d = 2-1 = 1
a_i = a + (i-1)(d)
= 1 + (i-1)(1)
= 1 + i - 1
= i

descending numbers

i	**OUTPUT**	**j**
1		0
2	1	1
3	2 1	2

AP -> 0, 1, 2
d = 1-0 = 1
a_i = a + (i-1)(d)
= 0 + (i-1)(1)
= i-1

```c
#include <stdio.h>
int main()
{
    int i, j, k;
    for (i = 1; i <= 3; i++)
    {
        // Print spaces
        for (j = 1; j <= (6 - 2*i); j++)
        {
```

```
            printf(" ");
        }
        // Print ascending numbers
        for (k = 1; k <= i; k++)
        {
            printf("%d ", k);
        }
        // Print descending numbers
        for (k = (i-1); k >= 1; k--)
        {
            printf("%d ", k);
        }
        printf("\n");
    }
    return 0;
}
```

f) Pattern: Diamond star pattern

```
    000000*
    0000* * *
    00* * * * *
    * * * * * * *
------------------------
    00* * * * *
    0000* * *
    000000*
```

spaces are taken as 0's (so its easier to count for calculating)

upper half spaces

i	OUTPUT	j	AP -> 6, 4, 2, 0
1	000000	6	d = 4-6 = -2
2	0000	4	a_i = a + (i-1)(d)
3	00	2	= 6 + (i-1)(-2)
4		0	= 6 - 2*i + 2
			= 8 - 2*i

lower half stars

i	OUTPUT	j	AP -> 1, 3, 5, 7
1	*	1	d = 3-1 = 2
2	* * *	3	a_i = a + (i-1)(d)
3	* * * * *	5	= 1 + (i-1)(2)
4	* * * * * * *	7	= 1 + 2*i - 2
			= 2*i - 1

```
For bottom half (descending):
i     spaces      j     AP -> 2, 4, 6 (for i = 3, 2, 1)
3     00          2         Using same formula: 8 - 2*i
2     0000        4           When i=3: 8-2*3 = 2
1     000000      6           When i=2: 8-2*2 = 4
                              When i=1: 8-2*1 = 6

i     OUTPUT      j     AP -> 5, 3, 1 (for i = 3, 2, 1)
3    * * * * *    5       Using same formula: 2*i - 1
2      * * *      3           When i=3: 2*3-1 = 5
1        *        1           When i=2: 2*2-1 = 3
                              When i=1: 2*1-1 = 1
```

```c
#include<stdio.h>
int main(){
    int i, j;

    // upper half rows (including middle)
    for (i = 1; i <= 4; i++){
        // upper half spaces
        for(j = 1; j <= 8 - i*2; j++){
                printf(" ");
        }
        // upper half stars
        for(j = 1; j <= 2*i - 1; j++){
                printf("* ");
        }
        printf("\n");
    }

    // lower half rows
    for (i = 3; i >= 1; i--){

        // lower half spaces
        for(j = 1; j <= 8 - i*2; j++){
                printf(" ");
        }
        // lower half stars
        for(j = 1; j <= 2*i - 1; j++){
                printf("* ");
        }
        printf("\n");
    }

    return 0;
}
```