Pattern Printing:

① n=5

```
for(i=1; i<=n; i++){
            1<=5
    for(j=1; j<=i; j++){
              5432 1<=1
        printf("*");
    }
    printf(" ");
}
```

i = 1, 2

j = 1, 2, 3, 4, 5 – * _ _ _ _ (2 3 4)

```
*
* *
* * *
* * * *
* * * * *
```

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

②

```
* * * * *
* * * *
* * *
* *
*
```

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

③ Filled Square.

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

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

④ Hollow Square (using 3 loops)

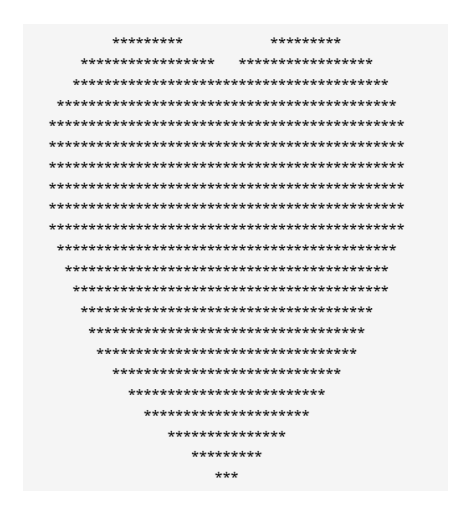
```
* * * * *
*       *
*       *
*       *
* * * * *
```

```
#include <stdio.h>
int main() {
   int n=5;
   for(int i=0;i<n;i++){
      printf("*");
   }
   printf("\n");

   for(int i=0;i<n-2;i++){
      printf("*");
      for(int j=0;j<n-2;j++){
         printf(" ");
      }
      printf("*");
      printf("\n");
   }

   for(int i=0;i<n;i++){
      printf("*");
   }
   return 0;
}
```

⑤ Big filled Heart:



```
#include <stdio.h>
int main() {
   float x,y;
   for(y=1.5;y>=-1.5;y-=0.1){
      for(x=-1.5;x<=1.5;x+=0.05){
         float a=x*x+y*y-1;
         if(a*a*a-x*x*y*y*y<=0){
            printf("*");
         }
         else{
            printf(" ");
         }
      }
      printf("\n");
   }
   return 0;
}
```

⑥ Empty Heart:

```
   *   *       *   *
*          *          *
*                     *
   *               *
      *         *
         *
```

```
#include <stdio.h>
int main() {
   for(int i=0;i<6;i++){
      for(int j=0;j<7;j++){
         if((i==0 && j%3!=0) || (i==1 && j%3==0) || (i-j==2) || (i+j==8)){
            printf("*\t");
         }else{
            printf(" \t");
         }
      }
      printf("\n");
   }

   return 0;
}
```

⑦ Zig-zag Pattern:

```
  *   *
 * * * *
*   *   *
```

```
#include <stdio.h>
int main() {
   for(int r=1;r<=3;r++){
      for(int c=1;c<=9;c++){
         if((r==1 && c%4==3) || (r==2 && c%2==0) || (r==3 && c%4==1)){
            printf("*");
         }
         else{
            printf(" ");
         }
      }
      printf("\n");
   }

   return 0;
}
```

Arrays:

→ collection of similar datatypes.

Declaration:

① arr[5] = {1,2,3,4,5};

② arr[] = {1,2,3,4,5,6};

③ arr[0] = 1;
   arr[1] = 2;   ⇒ {1,2,3};
   arr[2] = 3;

```
arr[5] = {1,2,3,4,5}
          0 1 2 3 4
n = 5, sum = 0;
for(i=0; i<n; i++){
   5432 1 0<5
   sum = sum + arr[i];
}      = 0+1 = 1 +2 = 3+3 = 6+4 = 10+5 = 15
printf("%d", sum);
}
```

Reverse an Array:

I/P: arr[] = {1, 2, 3, 4, 5}
              0 1 2 3 4

O/P: {5, 4, 3, 2, 1}

int rev[n]; →

rev[] = {5 4 3 2 1}
         0 1 2 3 4

```
for( i=0; i<n; i++){
    43 2 1 0<5
  rev[i] = arr[n-1-i];
}
```

rev[0] = arr[5-1-0] ⇒ arr[4]

rev[1] = arr[5-1-1] ⇒ arr[3]

rev[2] = arr[5-1-2] ⇒ arr[2]

rev[3] = arr[5-1-3] ⇒ arr[1]

rev[4] = arr[5-1-4] ⇒ arr[0]

arr[] = {1, 2, 3, 4, 5}
         0 1 2 3 4

rotated[] = {5 1 2 3 4}
             0 1 2 3 4

```
last = arr[n-1]
for(i=n-1; i>0; i--){
    4 0 3 2 1 >0
  rotated[i] = arr[i-1];
}
```

rotated[4] = arr[3]

rotated[3] = arr[2]

rotated[2] = arr[1]

rotated[1] = arr[0]