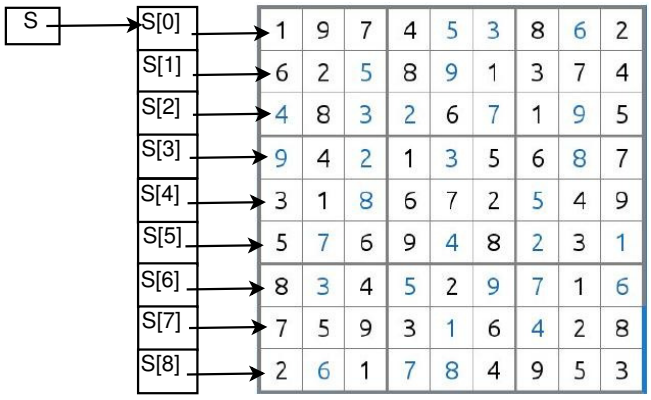
Q1. **Answer on this sheet only.** Consider the following program which implements a matrix as shown in the figure. It compiles without any error/warning. Answer the questions below. **Hint:** Go through the program and then use the figure to understand it (it may require some time). Most of the questions are independent of each other.

4+2+2+2+4+2+1+2+1 Marks

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

int main(void){
   int S[9][9] = {{1,9,7,4,5,3,8,6,2},{6,2,5,8,9,1,3,7,4},{4,8,3,2,6,7,1,9,5},
                  {9,4,2,1,3,5,6,8,7},{3,1,8,6,7,2,5,4,9},{5,7,6,9,4,8,2,3,1},
                  {8,3,4,5,2,9,7,1,6},{7,5,9,3,1,6,4,2,8},{2,6,1,7,8,4,9,5,3}};

   int i,j,k;
// Comment 1
   int* p=S[2];
   int* rowsum;
   rowsum = calloc(9,sizeof(int));

   int colsum[9]={0};
   int blocksum[9]={0};

   for(i=0;i<9;i++){
      for(j=0;j<9;j++){
         rowsum[i]=S[i][j]+rowsum[i];
      }
   }

   for(j=0;j<9;j++){
      for(i=0;i<9;i++){
         colsum[j]=S[i][j]+colsum[j];
      }
   }

   for(k=0;k<3;k++){
      for(i=0;i<3;i++){
         for(j=3*k;j<3*(k+1);j++){
            blocksum[k]=S[i][j]+blocksum[k];
            int i=0;                  // Be careful!!
            i=i+2;
         }
      }
   }

   for(i=0;i<5;i++){
      printf("%d %d %d\n",rowsum[i]-i,colsum[i]-i+1,blocksum[i]);
   }
   return 0;
}
```



Matrix S of size 9×9 in the adjacent program is initialized as shown in this figure. Note that each 2D array is nothing but an array of multiple 1D arrays. Hence, in the above figure, first row is 1D array `S[0]` (its elements are `S[0][0]` to `S[0][8]`), second row is 1D array `S[1]` (its elements are `S[1][0]` to `S[1][8]`), third row is 1D array `S[2]` (its elements are `S[2][0]` to `S[2][8]`) and so on. We also know that the name of an array is a pointer constant pointing to the first element of the array. `S` points to `S[0]`.

(a) What is the output of the program?

45 46 45
44 45 45
43 44 45
42 43 0
41 42 0

(b) `S[0]` points to `S[0][0]` (because S[0] is a pointer constant to the first element of this array which is `S[0][0]`), and `S[0]` contains value `3221210540`. What is the address of `S[0][1]` and `S[0][6]`?

3221210544, 3221210564

(c) If statement `int* p=S;` is inserted in place of `// Comment 1`. Will that be a valid statement in this program? Justify.

No. p is being redeclared. Even if p is not redeclared, dereference level incompatibility is there.

(d) If statement `char** p=S;` is inserted in place of `// Comment 1`. Will that be a valid statement in this program? Justify.

No. p is being redeclared. Even if p is not redeclared, dereference type incompatibility is there.

(e) Write the values of `*S[3]+1`, `*(S[0]+1)+2`, `**(S+2)` and `*p+2`.

10 11 4 6

(f) How much total memory space (in bytes) is utilized in storing the matrix `S`, as initialized in the program?

81*4 + 9*4 + 1*4 = 91*4 = 364 bytes

(g) How much heap (or dynamic) memory (in bytes) is used by the program?

9*4 = 36 bytes

(h) Release the memory allocated in heap at the appropriate place, i.e., do not release before its task is over.

Just before return 0; statement, write

```
free(rowsum);
rowsum=NULL;
```

(i) Is statement `colsum=p;` a valid statement just after the end of the first nested for loop? Justify.

No. colsum is a pointer constant. It cannot be assigned any value.