Day 3 - Material

Logic to print the given number pattern

555555555 i=5
544444445   i=4
543333345
543222345
543212345
543222345
543333345
544444445
555555555

Once you get acquainted with basics of number pattern printing, have a careful eye on to above pattern. Let me help you, to make things easier lets divide this entire pattern into two big parts and six smaller pieces. The first upper half contains three parts

```
---------    555555555    ---------
5--------    -4444444-    --------5
54-------    --33333--    -------45
543------    ---222---    ------345
5432-----    ----1----    -----2345
---------    ---------    ---------
---------    ---------    ---------
---------    ---------    ---------
---------    ---------    ---------
```

And the lower half of the pattern also contains three separate parts.

```
---------    ---------    ---------
---------    ---------    ---------
---------    ---------    ---------
---------    ---------    ---------
---------    ---------    ---------
5432-----    ----2----    -----2345
543------    ---333---    ------345
54-------    --44444--    -------45
5--------    -5555555-    --------5
```

Now, both upper and lower half's of the pattern would be printed separately in two separate outer loops. Considering the first upper half of the pattern. The logic to print upper half of the pattern:

1. To iterate through rows, initialize a loop from N till 1. (Note: We can also use loop from 1 to N but we have used it in decrementing order as the upper half is in decrementing order)
2. To print the first part of upper half, run an inner loop from N to 1 and print the current column number.
3. To print the second part of the upper half, run another inner loop from 1 to i*2-1 which is the total number of columns per row in this part. Inside this loop print the current row number.
4. To print the third part of the upper half, run another inner loop from current_row till N and print column number inside this loop.

Now, once you have printed the upper half of the pattern its time to get into lower half of the pattern. Logic to print the lower half of the pattern is

1. To iterate through rows, run a loop from 1 to N.
2. To print the first inner part of the lower half, run a loop from N to 1. Inside this loop print the column number.
3. To print the second inner part of the lower half, run another loop from 1 to i*2-1 which is the total number of columns per row in this part. Inside this loop print current row number + 1.
4. Finally to print the last inner part of lower half, run another loop from current row + 1 till N. Inside this loop print the current column number.

```
#include <stdio.h>
int main()
{
    int n, i, j;
    printf("Enter N: ");
    scanf("%d", &n); //n=5
```

```
// First upper half of the pattern
   for(i=n; i>=1; i--) //i=4 ;4>=1
   {
      // First inner part of upper half
      for(j=n; j>i; j--)  //j=5;5>4
      {
          printf("%d ", j);  //5
      }
      // Second inner part of upper half
      for(j=1; j<=(i*2-1); j++)  //i=4 j=1;j<=7
     {
          printf("%d ", i);       //4444444
      }
      // Third inner part of upper half
      for(j=i+1; j<=n; j++)   //j=5 5<=5
      {
          printf("%d ", j);
      }
      printf("\n");
   }
```

```
 // Second lower half of the pattern
    for(i=1; i<=n; i++)
    {
        // First inner part of lower half
        for(j=n; j>i; j--)
        {
            printf("%d ", j);
        }
        // Second inner part of lower half
        for(j=1; j<=(i*2-1); j++)
        {
            printf("%d ", i+1);
        }
        // Third inner part of lower half
        for(j=i+1; j<=n; j++)
        {
            printf("%d ", j);
        }
        printf("\n");
    }
    return 0;
}
```

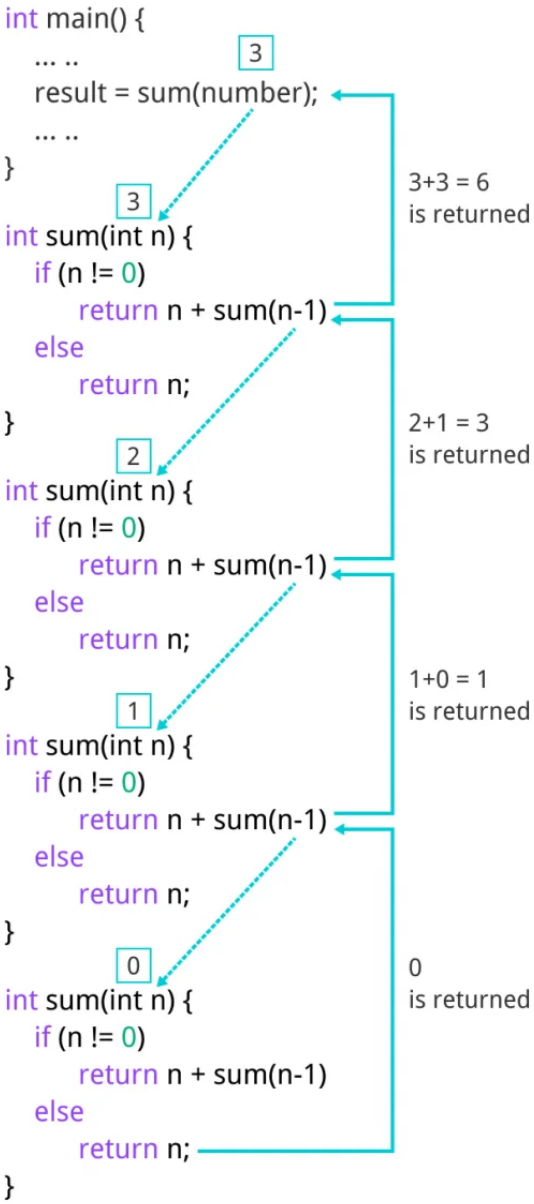
Recursive Function:

```
#include <stdio.h>

int factorial(int n) {
   //base case
   if(n == 0) {
      return 1;
   } else {
      return n * factorial(n-1);
   }
}
```

```
int fibbonacci(int n) {
   if(n == 0){
      return 0;
   } else if(n == 1) {
      return 1;
   } else {
      return (fibbonacci(n-1) + fibbonacci(n-2));
   }
}

n=5 fib(5) = fib(4)+fib(3) = 3+2 = 5
n=4 fib(4) = fib(3)+fib(2) = 2+1 = 3
n=3 fib(3) = fib(2)+fib(1) = 1+1 = 2
n=2 fib(2) = fib(1)+fib(0) = 1+0 = 1
n=1 fib(1) = fib(1) = 1
int main() {
   int n = 5;
   int i;

   printf("Factorial of %d: %d\n" , n , factorial(n));
   printf("Fibbonacci of %d: " , n);

   for(i = 0;i<n;i++) {
      printf("%d ",fibbonacci(i));
   }
}

Output:
Factorial of 5: 120
Fibbonacci of 5: 0 1 1 2 3
```

Sum of Natural Numbers Using Recursion

```
#include <stdio.h>
int sum(int n);

int main() {
```




```
    int number, result;

    printf("Enter a positive integer: ");
    scanf("%d", &number);

    result = sum(number);

    printf("sum = %d", result);
    return 0;
}
```

```
int sum(int n) {
    if (n != 0)
        // sum() function calls itself
        return n + sum(n-1);
    else
        return n;
}
```

Output:

```
Enter a positive integer:3
sum = 6
```

Call by value and Call by reference

```
#include<stdio.h>
void change(int num) {
    printf("Before adding value inside function num=%d \n",num);
    num=num+100;
    printf("After adding value inside function num=%d \n", num);
}
int main() {
    int x=100;
    printf("Before function call x=%d \n", x);
    change(x);//passing value in function
    printf("After function call x=%d \n", x);
return 0;
}
```

Output:

```
Before function call x=100
Before adding value inside function num=100
After adding value inside function num=200
After function call x=100
```

```
#include <stdio.h>
void swap(int, int); //prototype of the function
int main()
{
   int a = 10;
   int b = 20;
   printf("Before swapping the values in main a = %d,
b = %d\n",a,b);
   swap(a,b);
   printf("After swapping values in main a = %d, b = %d\n",a,b);
}
void swap (int a, int b)
{
   int temp;
   temp = a;
   a=b;
   b=temp;
    printf("After swapping values in function a = %d, b = %d\n”,a,b);
}
```

Output:
Before swapping the values in main a = 10, b = 20
After swapping values in function a = 20, b = 10
After swapping values in main a = 10, b = 20

Call by reference in C

- In call by reference, the address of the variable is passed into the function call as the actual parameter.
- The value of the actual parameters can be modified by changing the formal parameters since the address of the actual parameters is passed.
- In call by reference, the memory allocation is similar for both formal parameters and actual parameters. All the operations in the function are performed on the value stored at the address of the actual parameters, and the modified value gets stored at the same address

```
#include<stdio.h>
void change(int *num) {
    printf("Before adding value inside function num=%d \n",*num);
    (*num) += 100;
    printf("After adding value inside function num=%d \n", *num);
}
int main() {
    int x=100;
    printf("Before function call x=%d \n", x);
    change(&x);
    printf("After function call x=%d \n", x);
return 0;
}
```

Output:

```
Before function call x=100
Before adding value inside function num=100
After adding value inside function num=200
After function call x=200
```

Call by reference Example: Swapping the values of the two variables

```
#include <stdio.h>
void swap(int *, int *); //prototype of the function
int main()
{
    int a = 10;
    int b = 20;
    printf("Before swapping the values in main a = %d, b = %d\n",a,b)
;
```

```
    swap(&a,&b);
    printf("After swapping values in main a = %d, b = %d\n",a,b);
}
void swap (int *a, int *b)
{
    int temp;
    temp = *a;
    *a=*b;
    *b=temp;
    printf("After swapping values in function a = %d, b = %d\n",*a,*b);

}
```

Output
Before swapping the values in main a = 10, b = 20
After swapping values in function a = 20, b = 10
After swapping values in main a = 20, b = 10

4. Write a C program to calculate the distance between the two points.
Note: x1, y1, x2, y2 are all double values.
Formula:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

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

int main() {
        float x1, y1, x2, y2, gdistance;
        printf("Input x1: ");
        scanf("%f", &x1);
        printf("Input y1: ");
        scanf("%f", &y1);
             printf("Input x2: ");
```

```
        scanf("%f", &x2);
        printf("Input y2: ");
        scanf("%f", &y2);
        gdistance = ((x2-x1)*(x2-x1))+((y2-y1)*(y2-y1));
        printf("Distance between the said points: %.4f",
sqrt(gdistance));
        printf("\n");
        return 0;
}

#include <stdio.h>
#include <math.h>
#include<string.h>
#include<stdlib.h>
int main(){
    int n,odd=0,even=0,len,rem,flag;
    char str1[20];
    scanf("%d",&n);
    sprintf(str1,"%d",n); //convert int to string
    len=strlen(str1);
    if(len%2==0){ //123456
        flag=0;
        while(n!=0){
            if(flag==0){
                rem=n%10;
                even+=rem;
                flag=1;
            }
            else{
                rem=n%10;
                odd+=rem;
                flag=0;
            }
            n=n/10;
        }
    }
```

```
    else{
        flag=1; //12345
        while(n!=0){
            if(flag==0){
                rem=n%10;
                even+=rem; //even=4+2
                flag=1;
            }
            else{
                rem=n%10;
                odd+=rem; //odd=5+3+1
                flag=0;
            }
            n=n/10;
    } }
    printf(“%d",abs(even-odd));
    return 0; }
```

Example 1: C Program to Convert Binary Number to Decimal

```
// convert binary to decimal

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

// function prototype
int convert(long long);

int main() {
  long long n;
  printf("Enter a binary number: ");
  scanf("%lld", &n);
  printf("%lld in binary = %d in decimal", n, convert(n));
  return 0;
}

// function definition
int convert(long long n) {
```

```
  int dec = 0, i = 0, rem;

  while (n!=0) {
    rem = n % 10;
    n /= 10;
    dec += rem * pow(2, i);
    ++i;
  }

  return dec;
}
```

Output:
Enter a binary number: 1101
1101 in binary = 13 in decimal

Example 2: C Program to convert decimal number to binary

```
// convert decimal to binary

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

long long convert(int);

int main() {
  int n, bin;
  printf("Enter a decimal number: ");
  scanf("%d", &n);
  bin = convert(n);
  printf("%d in decimal =  %lld in binary", n, bin);
  return 0;
}

long long convert(int n) {
  long long bin = 0;
  int rem, i = 1;
```

```
  while (n!=0) {
    rem = n % 2;
    n /= 2;
    bin += rem * i;
    i *= 10;
  }

  return bin;
}
```

Program to Convert Binary to Octal

```
#include <math.h>
#include <stdio.h>
int convert(long long bin);
int main() {
    long long bin;
    printf("Enter a binary number: ");
    scanf("%lld", &bin);
    printf("%lld in binary = %d in octal", bin, convert(bin));
    return 0;
}

int convert(long long bin) {
    int oct = 0, dec = 0, i = 0;

    // converting binary to decimal
    while (bin != 0) {
        dec += (bin % 10) * pow(2, i);
        ++i;
        bin /= 10;
    }
    i = 1;
```

```
    // converting to decimal to octal
    while (dec != 0) {
        oct += (dec % 8) * i;
        dec /= 8;
        i *= 10;
    }
    return oct;
}
```

Output:

Enter a binary number: 101001
101001 in binary = 51 in octal

Program to Convert Octal to Binary

```
#include <math.h>
#include <stdio.h>
long long convert(int oct);
int main() {
    int oct;
    printf("Enter an octal number: ");
    scanf("%d", &oct);
    printf("%d in octal = %lld in binary", oct, convert(oct));
    return 0;
}

long long convert(int oct) {
    int dec = 0, i = 0;
    long long bin = 0;

    // converting octal to decimal
    while (oct != 0) {
        dec += (oct % 10) * pow(8, i);
        ++i;
        oct /= 10;
    }
    i = 1;
```

```
   // converting decimal to binary
    while (dec != 0) {
        bin += (dec % 2) * i;
        dec /= 2;
        i *= 10;
    }
    return bin;
}
```

Output:

Enter an octal number: 67
67 in octal = 110111 in binary

Program to Convert Decimal to Octal

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

// function prototype
int convertDecimalToOctal(int decimalNumber);

int main() {

    int decimalNumber;

    printf("Enter a decimal number: ");
    scanf("%d", &decimalNumber);

    printf("%d in decimal = %d in octal", decimalNumber,
convertDecimalToOctal(decimalNumber));

    return 0;
}

// function to convert decimalNumber to octal
int convertDecimalToOctal(int decimalNumber) {
```

```
    int octalNumber = 0, i = 1;

    while (decimalNumber != 0) {
        octalNumber += (decimalNumber % 8) * i;
        decimalNumber /= 8;
        i *= 10;
    }

    return octalNumber;
}
```

Output:

```
Enter a decimal number: 78
78 in decimal = 116 in octal
```

Program to Convert Octal to Decimal

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

// function prototype
long long convertOctalToDecimal(int octalNumber);

int main() {

    int octalNumber;

    printf("Enter an octal number: ");
    scanf("%d", &octalNumber);

    printf("%d in octal = %lld in decimal", octalNumber,
convertOctalToDecimal(octalNumber));

    return 0;
}

// function to convert octalNumber to decimal
```

```
long long convertOctalToDecimal(int octalNumber) {
    int decimalNumber = 0, i = 0;

    while(octalNumber != 0) {
        decimalNumber += (octalNumber%10) * pow(8,i);
        ++i;
        octalNumber/=10;
    }

    i = 1;

    return decimalNumber;
}
```

Output:

Enter an octal number: 116
116 in octal = 78 in decimal

4. Write a program where a user-defined function coverts a binary number to decimal and then calls another user-defined function which converts the decimal number into an octal number.
Sample input: 1000
Sample output: 10.

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

// function prototype
int convert(int);
int convertDecimalToOctal(int decimalNumber);

int main() {
  int n,result;
  printf("Enter a binary number: ");
  scanf("%d", &n);
  result=convert(n);
```

```
  printf("%d",convertDecimalToOctal(result));
  return 0;
}

// function definition
int convert(int n) {
  int dec = 0, i = 0, rem;

  while (n!=0) {
    rem = n % 10;
    n /= 10;
    dec += rem * pow(2, i);
    ++i;
  }

  return dec;
}
int convertDecimalToOctal(int decimalNumber) {
    int octalNumber = 0, i = 1;

    while (decimalNumber != 0) {
        octalNumber += (decimalNumber % 8) * i;
        decimalNumber /= 8;
        i *= 10;
    }

    return octalNumber;
}
```

5. Write a program using recursive function to find the nth term of the given series. 3, 7, 11, 15, 19, 23,...
Sample input: Enter the value of n: 10
Sample output: The 10^{th} term is: 39

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

```
int recur(int);
int main(){
    int n,res;
    scanf("%d",&n);
    res=recur(n);
    printf("%d",res);
    return 0;
}
int recur(int n){
    if(n==1){
        return 3;
    }
    else{
        return 4+recur(n-1);
    }
}
```