Assignment No. 2

1) Print element of an array forward & backward.

Example 1:
Input: arr = [1, 2, 3, 4]
output (forward): 1 2 3 4
output: (Backward): 4 3 2 1

→ Int [] arr = { 1, 2, 3, 4 };
~~int~~ for (int i = ~~10~~ 0; i<5; i++) {
print (i);
}
~~or~~ output: 1, 2, 3, 4

backward

for (int i = 4; i>0; i--) {
print (i);
}

output: 4, 3, 2, 1

2) Find sum of all ~~produ~~ digits of a Number

Example 2:
Input N: 1234
output: 10

Explanation = 1+2+3+4 = 10

Parameterized way

main() { int sum = 0;
Int sum = f(n);
print (sum);

Here n is 4

```
int f(int n) {
    if (n == 0)
        return 0;
    Return n + f(n-1);
}
```

4+3=7

```
int f(int n) {
    if (n == 0) {
        return 0;
    }
    Return n + f(n-1);
}
```

3+2=5

```
Int f(int n) {
    if (n == 0) {
        return 0;
    }
    Return n + f(n-1);
}
```

2+1=3

```
Int f(int n) {
    if (n == 0) {
        return 0;
    }
    Return n + f(n-1);
}
```

1+0=1

```
Int f(int n) {
    if (n == 0) {
        return 0;
    }
    Return n + f(n-1);
}
```

0+0=0

```
Int f(int n) {
    if (n == 0) {
        return 0;
    }
```

Q.2) find sum all digits of a Number

Example:

Input: N: 1234

output 10: 10

functional way

N = 4

```
Main () {
   Int sum = f(n);
     print( sum);
}

Int f ( int n) {
    If ( n == 0) {
       return 0;
    }
   Return n + f(n-1);
}
```

4+6=10 4 3-1=3

2

```
Intf (int n) {
 If (n == 0) {
    return 0;
 }
 return n + (n-1);
}
```

2+1 1-1=0 1

1+0=3

1

0

```
Int f (int n)
 If (n == 0) {
     0 = 0
    return 0;
}
```

3

```
Intf (int n) {
 If (n == 0) {
    return 0
 }
 Return n + (n-1);
}
```

3-1=2

3+3=6

1

```
Inf ( int n) {
 If (n == 0) {
    return 0;
 }
 return n + (n-1
}
```

1-1=

1+0 = 1

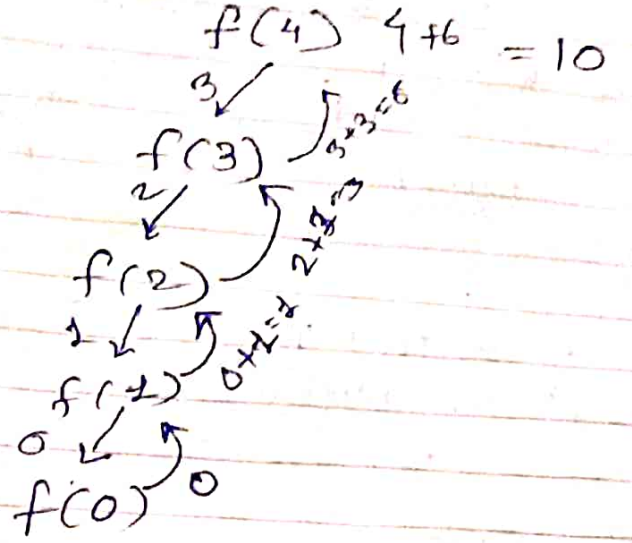
f(0)

f(1)

f(2)

f(3)

f(4)



Parameterized way

Here n = 4

```
Main () {
   int sum = 0;
    f ( sum, n);
}
```

```
void f ( sum, n) {        (sum: 0, n: 4)
 If ( n<1) {              (4<1) false
Print(sum); return;
   }
    sum = sum + n;        0+4 = 4
     f ( sum, n-1);
```

↳

```
void f ( sum, n-1) {      (sum: 4, n-1: 3)
      If (n<1) {          (3<1) false
Print (sum); return;
      }
    sum = sum + n         = 4+3 = 7
     f = (sum, n-1);
}
```

7 2

```
Void f (sum, n-1) {
    If ( n<1 ) {          (2<1)
print(sum);   return;      false
    }
```

```
sum = sum+n          7+2 = 9
f (sum, n-1);
}
```

9 1

```
void f ( sum, n-1) {
    If ( n<1) {          (1<1)
      print (sum);       false
       return;
    }
sum= sum+n          9+1 = 10
   f (sum, n-1);
}
```

10 0

```
void f ( sum, n-1) {
   If (n<1) {
      print (sum);
       return;
   }
```

~~Pat~~ Output: 10

f(10,0)
f(9,1)
f(7,2)
f(4,3)
f(0,4)

f(0,4)

4,3

f(4,3)

7,2

f(7,2)

9,1

f(9,1)

10,0

3) find the product of digits of Number using Recursion

Example 1:

Input : N = 231

output : 6

Explanation: 2×3×1 = 6

parameterized way

Here N = 231

```
Main () {
        f (int n) {          // 231
            if (n == 0) {
                Return 1;
            }
            return (n%10);
            prod = (n%10);          231%10 = 1
            Return prod * f(n/10);
                  1          23
        }
```

```
        f (int n) {          // 23
            if (n == 0) {          23 == 0
                Return 1;          false
            }
            prod = (n%10);          23%10 = 3
            Return Prod * f(n/10);
                  3   *      2
        }
```

```
        f (int n) {          // 2
            if (n == 0) {          (2 == 0)
                return 1;          false
            }
            Prod = (n%10) =          2%10 = 2
            Return Prod * f(n/10);
```

```
f(int n) {
    if (n == 0) {
        Return 1;
    }
}
```

0 == 0
true
Return 1

4. Count Number of digits of Number

Example 1:
Input N: 98765
Output: 5

```
main () {
    f(int n)
    int count;;
    count
}
    Intf (int n) {
        If (n==0) {
            return 0;
        }
```

98765

98765==0

98765/10 = 9876

```
count= return 1 + f(n/10);
    print (count);
    count = return 1+ f(n/10);
        print ( count);
}
```

1+4=5

9876

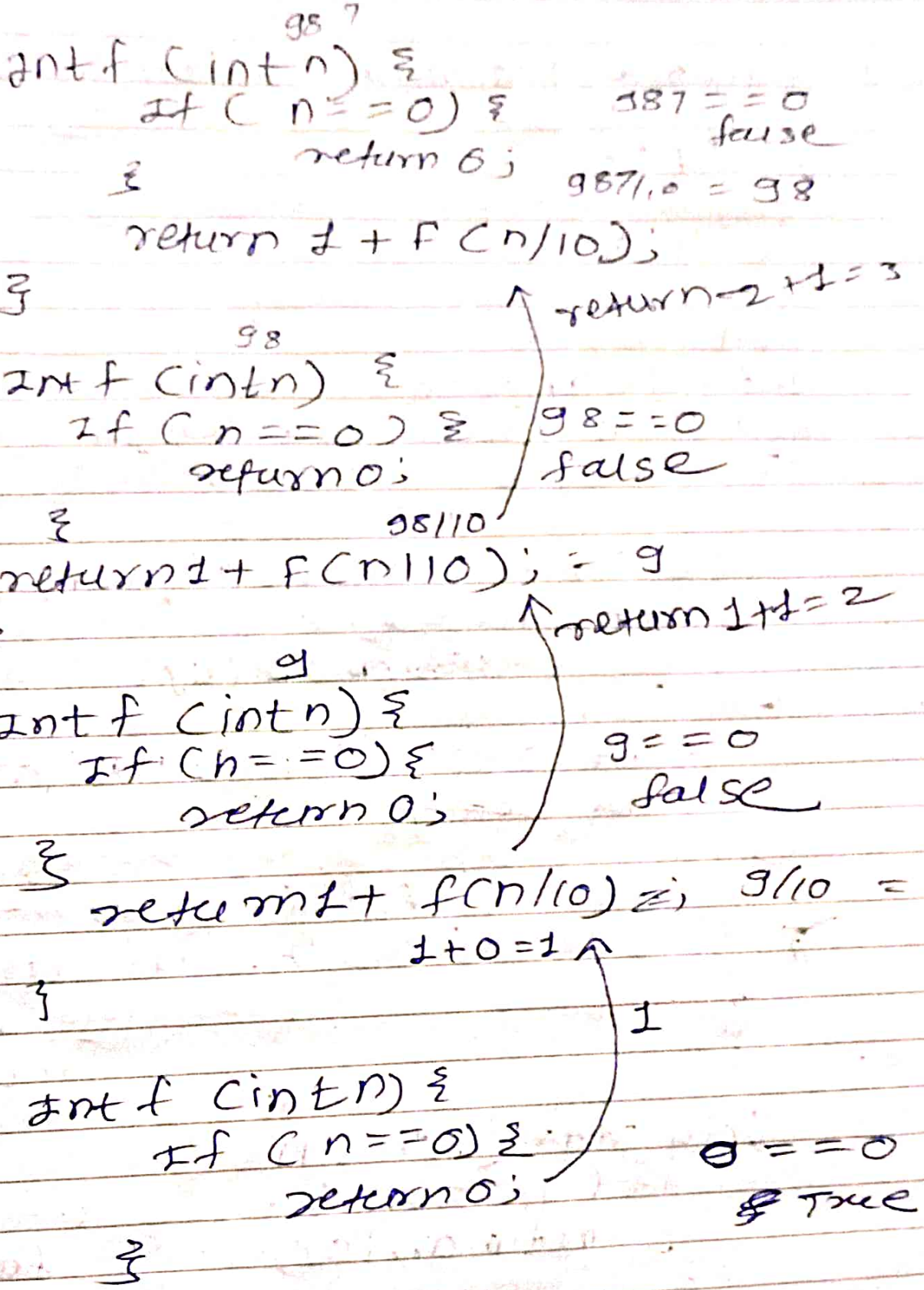
```
Intf ( int n ) {
    If (n==0) {
        return 0;
    }
```

9876==0
false

9876/10 = 987

```
return 1 + f (n/10);
```

Count= 0 1+3=4


987
int f (int n) {
if (n == 0) {
return 0;
}
return 1 + f (n/10);
}
987 == 0
false
987/10 = 98
return 2+1 = 3
98
int f (int n) {
if (n == 0) {
return 0;
}
return 1 + f (n/10);
}
98 == 0
false
98/10
return 1+1 = 2
9
int f (int n) {
if (n == 0) {
return 0;
}
return 1 + f (n/10);
}
9 == 0
false
9/10 = 0
1+0 = 1
1
int f (int n) {
if (n == 0) {
return 0;
}
}
0 == 0
True

5) Find the maximum element in an array

→ Example 1:

Input: arr [2, 5, 9, 1, 6]

output: 9

```
Main () {
int [] arr = [2, 5, 9, 1, 6]
int m = Max (arr, arr.length) ;
}   s.o.p ( Maximum element is +m
```

```
Void Max ( int [] arr, int n ) {
    if (n == 1) {                    5 == 1
        return arr[0];               false
    }
    int Max (int [] arr, n-1);
    int return
}
                                     5-1 = 4
int a = Max (arr, n-1);
return Math.max (arr [n-1],
                 9, 6
                 max = 6
}
```

9

```
                                     4
Void Max ( arr, n ) {
    if (n == 1) {                    4 == 1
        retur arr[0];                false
    }
                                     4-1 = 3
int a = Max (arr, n-1);
return Math.max (arr[n-1], a);
                 9, 1
```

```
Void Max (arr, n) {
  if (n == 1) {
    return arr[0];
  }
  int a = Max (arr, n-1);
  Return math max (arr[n-1], a);
}
```

```
Void max (arr, n) {
  if (n == 1) {
    return arr[0];
  }
  int a = Max (arr, n-1);
  Return math. max (arr[n-1], a);
}
```

```
Void max (arr, n) {
  if (n == 1) {
    return arr[0];
  }
}
```

6) Check of an array is sorted (strictly increasing)

- Example 1:

Input : arr = [1, 2, 3, 4]

output : true

Example 2:
Input : arr = [1, 2, 2, 3]
false
~~Input : arr = [1, 2, 2, 3]~~
~~Output:~~

```
Main () {
    int [] arr = [1, 2, 3, 4];
    S.o.p ("array is strickly
          increasing" + f(arr, arr.
}

boolean f (int[] arr, int n) {
    if (n == 1) {
        return true;
    }

    if ( arr[n-2] >= arr[n-1])
        return false;
    }
    return f(arr, n-1);
}

boolean f (int[] arr, n) {
    if (n == 1) {
        return true;
    }

    if (arr[n-2] >= arr[n-1])
        return false;
    }
```

```
boolean f(arr, n-2) {
    if (n==1e) {
        return true;
    }
    if (arr[n-2] >= arr[n-1]) {
        return false;
    }
    return f(arr, n-1);
}
gth));

boolean f(arr, n-1) {
    if (n==1) {
        return true;
    }
}
```

2 == 0 false

1 >= 2 True

7. Check if a Number is prime

Example1:
Input N = 2
Output: true

Example2:-
Input N = 10
output false

~~Main () {~~
~~int n = 10;~~
~~prime~~
~~}~~

~~bott~~

```
main () {
    int num = 10;
    If (n <= 1) {
        is prime =
        return false;
    }
    for (int i = 2; i < n; i++ {
        If (n % i == 0) {
            return false;
            break;
        }
    }
    if (return false) {
        S.O.p (n+" is a prime ");
    } else {
        S.O.p (n+" is a not prime");
    }
}
```

8 Reverse a number using recursion.

Example 1:

Input N = 1234

output = 4321

```
Main () {
    int num = 1234;
    int reversed = reverse (N, 0);
    S.O.p ("Reversed number:" + reversed)
}

int reverse (int n, int rev) {
```

1234 0

```
if (n==0) {                    1234==0
} return rev;                  false
return reverse(n/10, rev*10+n%10);
                               1234/10   0*10+1234%10
                               123       0+4=4
}
int reverse (int n, int rev) {
    if (n==0) {                123==0
        return rev;            false
    }
    return reverse(n/10, rev*10+n%10);
                               123/10 , 4*10+123%10
                               12        40+3=43
                               12        +43
}
int reverse (int n, int rev) {
    if (n==0) {                12==0
        return rev;            false
    }
    return reverse(n/10, rev*10+n%10);
                               12/10 , 43*10+2%10
                               1         430+2=432
                                         432
}
int reverse (int n, int rev) {
    if (n==0) {                1==0
        return rev;            false
    }
    return reverse(n/10, rev*10+n%10);
                               1/10      432*10+n%10
                               0         432+1%10
                                         4321
}
int reverse (int n, int rev) {
    if (n==0) {                0==0
        return rev;            True
```

9. check if a number is palindrome

Example 1:

Input N = 121

Output : true

Input N = 123

output: false

```
Main () {
  int num = 121;
  int reversed = reverse (num, 0);
  if (num = reversed {
      s.o.p ( ~~Print~~ " Number is palindr
  } else {
      s.o.p ("Number is not palindrom
  }
}
```

121 0

```
int ~~void~~ reverse (int n, int rev) {
    if (n == 0) return rev;        121==0
  }
  return revers ( n/10, rev*10 + n%
}                  121/10   0*10+12%1
                    12        2
                   12
```

```
~~Void~~ int reverse (int n, int rev) {
    if (n == 0) return rev;        12==
}                                  false
```

return reverse (n/10, rev*10 + n%

12/10 = 1 1*10 + 2 =

```
int reverse (int n, int rev){
  if (n==0) return rev;
}
return reverse(n/10, rev*10+n%10);
}
int reverse (int n, int rev){
  if (n==0) return rev;
```

Number is palindrome.

10) Find GCD (HCF) of two Numbers using recursion.

Example :

input : A = 24, B = 36

output : 12

```
main () {
  int a = 24;
  int b = 36;
  int result = f(A,B);
  S.O.P("GCD is "+result);
}
int f( int a, int b) {
  if (b==0) {
    return a;
  }
  return f(b, a%b);
}
int f(int a, int b) {
  if (b==0) {
    return a;
```

```
    return f(b, a%b);
}
int f(int a, int b) {
    if (b==0) {
        return a;
    }
    return f(b, a%b);
}
int f(int a, int b) {
    if (b==0) {
        return a;
    }
}
```

output: GCD is 12

ii) print all Numbers from 1 to N divisible by 3

Example 1:

Input N=10

output: 3.6.9

```
Main() {
    int n=10;
    f(int 1, n);
}
Void f(int temp, int n) {
    if (temp>n) {
        return;
    }
```

1 % 3 ≠ 0

```
if (temp%3==0){
    s.o.p(~~curr~~temp);
}
~~f (temp+1, n);~~
f(temp+1, n);
}
```

```
void f(int temp, int n){
    if (temp>n){
        return;
    }
    if (temp%3==0){
        s.op(temp);
    }
    f(temp+1, n);
}
```

```
void f(int temp, int n){
    if (temp>n){
        return;
    }
    if (temp%3==0){
        s.o.p(temp);
    }
    f(temp+1, n);
}
```

Now it iterate till temp is Reach to ~~10~~ 11 so

12) Find power of a number using recursion

Examples:

Input : A = 2, B = 4

Output : 16

Explanation : 2^4 = 2x2x2x2 = 16

```
Main () {
  int A = 2;
  int B = 4;
  int result = power(A,B);
  S.O.p ("the result is" + result);
}

int power (int A, int B) {
  if (B == 0) {
    return 1;
  }
  return A * power (A, B-1);
}

int power (int A, int B) {
  if (B == 0) {
    return 1;
  }
  return A * power (A, B-1);
}

int power (int A, int B) {
  if (B == 0) {
    return 1;
  }
  return A * power (A, B-1);
}
```

4 == 0 false

2 * 2 = 4

3 == 0 false

4 * 2 = 8

2 == 0 false

2 * 2 = 4

1

2

3

4

```
int Power (int A, int B) {
if (B == 0) {
    return 1;
}
return A * Power (A, B-1);
}
```

```
int Power (int A, int B) {
if (B == 0) {
    return 1;
}
```

The Result is 1c

(3) count how many times a digit appear in a Number

Example 1:

Input: N = 717237, D = 7

output: 3

```
Main () {
    int N = 717237
    int D = 7
int result = f(N, D);
    S.O.P (result);
}
void f (int N, int D) {
    if (N == 0) {
        return 0;
    }
int last = n % 10;
if (last == D) {
    return 1 + f (N/10, D);
}
```

```
else {
    return f (N/10, D);
  }
}
```

```
                71723   7
void f (int N, int D) {
  if (N == 0) {          71723 == 0
    return 0;                false
  }

  int last = n%10;       71723%10 = 3
  If (last == D) {                false
    return 1 + f (N/10, D);
  } else {
    return f (N/10, D);
                         71723/10 = 7172, 7
  }
}
```

```
                71723   7
void f (int N, int D) {
  if (N == 0) {          7172 == 0
    return 0;                false
  }
  int last = n%10;       7172%10 = 2
  If (last == D) {                false
    return 1 + f (N/10, D);
  } else {
    return f (N/10, D);
                         7172/10 = 717   7
  }
}
```

```
                717
void f (int N, int D) {
  if (N == 0) {
    return 0;
  }
```

find the first index of an element in an array

Example 1:

Input: arr=[4, 2, 7, 7, 9], key = 7

output: 2

```
Main () {
 int [] arr = { 4, 2, 7, 7, 9};
 int key = 7;

 int result = f ( arr, 0, key);
   S.O.P ("First index of " +key+
          " is " +result);
}

   int f ( int [] arr, index, int key);
     if (index == arr.length) {
         return -1;
     }

     if (arr [index] == key) {
         return index;
     }
     return f (arr, indx+1, key);
   }

   intf (                    ); {
     if (index == arr.length) {
         return -1;
     }
   If ( arr [index] = key) {
       return index;
   }
```

0

0 == 4 false

4 == 7 false

1

1 == 4 false

1 == 7 false

return f (arr, index+1, key);
}

int f (
if (index == arr.length) { 2==5
return -1; false
}

If (arr[index] = key) { 7==7
return index;
}
first index of 7 is 2

Q.15) find the last index of an element in an array

Example:
Input: arr = [4, 2, 7, 7, 9]
key = 7
output: 3;

Main () {
int[] arr = {4, 2, 7, 7, 9};
int key = 7;
int result = f (arr, 0, key);
s.o.p ("Last index of" + key + "is" + result);
)
}

int f (int [] arr, int index, int key) {
4 7
if (index = arr.length) { 4 == 7
return -1; false
}

```
int a = f(arr, index+1, key);
if(a != -1){
    return a;
}
if(arr[index] == key){
    return index;
}
return -1;
}
f(arr, index, key){
    if(index == arr.length){
        return -1;
    }
int a = f(arr, index+1, key);
```

index == arr.length → false