GATE

DS&AI

Python-For Data Science

Functions

DPP: 1

Q1 The return value of below function f(7) is _____

```
def f(x):
    if x<=1:
        return x-1
    return x + f(x-2)
```

Q2 The output printed by below code is _____

```
def fun(i,j):
    if i==j:
        print(i+j,end=" ")
    else:
        print(i-1,j,end=" ")
        fun(i-2,j+2)
fun(12,0)
```

(A) 12 0 10 2 8 4 12

(B) 11 1 9 3 7 5 12

(C) 11 0 9 2 7 4 12

(D) 12 1 10 3 8 5 12

Q3 The output printed by below code segment is _____

```
def function(i):
    if i<=0:
        return 0
    print(i-1,end=' ')
    function(i-2)
    print(i+1,end=' ')
function(5)
```

(A) 5 3 1 3 5 7

(B) 4 2 0 2 4 6

(C) 5 3 1 2 4 6

(D) 5 3 1 1 3 5

Q4 Consider the below code:

```
def f(i):
    count =1
    if i <= 0:
        return
    for x in range(i):
        k = count + g(x + 1)
        count = count + x
    return k
def g(i):
    j = 1
    if i < 1:
        return i + 1
    for x in range(i + 1):
        j = j + x
    return j
```

The return value of f(5) is _____

Q5 The output printed by below code is _____

```
def f(i):
    if len(i)==0:
        return
    else:
        i[-1]=i[0]
        j=i[:-2]
        print(j)
        f(j[3:])
a=[11,23,34,45,56]
f(a)
```

(A) [11,23]

(B) [11,23,34]

(C) [11,23,34,45]

(D) [11,23,34,45,56]

Q6

```
def fun(n):
    x = 1
    if n == 1:
        return x
    for k in range(1, n):
        x += fun(k) * fun(n - k)
    return x
```

The return value of above code on f(3) call is _____

Q7 The output printed is _____

```
def foo(n: int, r: int):
    if n > 0:
```

```
        return (n % r) + foo(n // r, r)
    else:
        return 0
print(foo(513,2))
```

(A) 0 (B) 1

(C) 2 (D) 9

Q8 The output of below python code segment is

_____

```
def fun(x):
    if x > 0:
        x=x-1
        fun(x)
        print(x, end='')
        fun(x)
        x=x-1
fun(3)
```

(A) 0102010 (B) 2012010

(C) 1010202 (D) 0101010

GATE

Answer Key

Q1	15	Q5	(B)
Q2	(C)	Q6	5
Q3	(B)	Q7	(C)
Q4	23	Q8	(A)

Hints & Solutions

Q1 Text Solution:

Let's break down the evaluation of `f(7)`:

1. **Call `f(7)`**:
   - Since $7 > 1$7>1, the function proceeds to `return 7 + f(7 - 2)`.
   - So, it returns $7 + f(5)$7+f(5).
2. **Call `f(5)`**:
   - Since $5 > 1$5>1, the function proceeds to `return 5 + f(5 - 2)`.
   - So, it returns $5 + f(3)$5+f(3).
3. **Call `f(3)`**:
   - Since $3 > 1$3>1, the function proceeds to `return 3 + f(3 - 2)`.
   - So, it returns $3 + f(1)$3+f(1).
4. **Call `f(1)`**:
   - Since $1 \leq 1$1≤1, the function proceeds to `return 1 - 1`.
   - So, it returns `0`.

Now we can substitute back up the chain:

- **Return value of `f(3)`**:
  - $f(3) = 3 + f(1)$f(3)=3+f(1)
  - $f(1) = 0$f(1)=0
  - So, $f(3) = 3 + 0 = 3$f(3)=3+0=3.
- **Return value of `f(5)`**:
  - $f(5) = 5 + f(3)$f(5)=5+f(3)
  - $f(3) = 3$f(3)=3
  - So, $f(5) = 5 + 3 = 8$f(5)=5+3=8.
- **Return value of `f(7)`**:
  - $f(7) = 7 + f(5)$f(7)=7+f(5)
  - $f(5) = 8$f(5)=8
  - So, $f(7) = 7 + 8 = 15$f(7)=7+8=15.

Therefore, the return value of `f(7)` is **15**.

Q2 Text Solution:

1. **First Call: `fun(12, 0)`**
   - Since `i != j` (12 != 0), the `else` branch is executed.
   - Print `12 - 1` and `0` which results in `11 0`.
   - Recursively call `fun(12 - 2, 0 + 2)`, i.e., `fun(10, 2)`.
2. **Second Call: `fun(10, 2)`**
   - Since `i != j` (10 != 2), the `else` branch is executed.
   - Print `10 - 1` and 2 which results in `9 2`.
   - Recursively call `fun(10 - 2, 2 + 2)`, i.e., `fun(8, 4)`.
3. **Third Call: `fun(8, 4)`**
   - Since `i != j` (8 != 4), the `else` branch is executed.
   - Print `8 - 1` and 4 which results in `7 4`.
   - Recursively call `fun(8 - 2, 4 + 2)`, i.e., `fun(6, 6)`.
4. **Fourth Call: `fun(6, 6)`**
   - Since `i == j` (6 == 6), the `if` branch is executed.
   - Print `6 + 6` which results in 12.

Putting it all together, the output will be:

- From the first call: `11 0`
- From the second call: `9 2`
- From the third call: `7 4`
- From the fourth call: `12`

So, the complete output sequence is:
`11 0 9 2 7 4 12`
The correct option is **C**.

Q3 Text Solution:

1. **First Call: `function(5)`**
   - Since `5 > 0`, it prints `5 - 1`, which is 4.
   - Then it makes a recursive call to `function(5 - 2)`, i.e., `function(3)`.
2. **Second Call: `function(3)`**
   - Since `3 > 0`, it prints `3 - 1`, which is 2.

- Then it makes a recursive call to `function(3 - 2)`, i.e., `function(1)`.

3. **Third Call: `function(1)`**
   - Since `1 > 0`, it prints `1 - 1`, which is `0`.
   - Then it makes a recursive call to `function(1 - 2)`, i.e., `function(-1)`.
4. **Fourth Call: `function(-1)`**
   - Since `-1 <= 0`, it returns `0` without printing anything.

After the recursive calls complete, the function returns to the previous call:

- **Returning to `function(1)`**:
  - After the recursive call `function(-1)` completes, it prints `1 + 1`, which is 2.
- **Returning to `function(3)`**:
  - After the recursive call `function(1)` completes, it prints `3 + 1`, which is 4.
- **Returning to `function(5)`**:
  - After the recursive call `function(3)` completes, it prints `5 + 1`, which is 6.

Combining all these outputs:

- From `function(5)`: prints 4, calls `function(3)`.
- From `function(3)`: prints 2, calls `function(1)`.
- From `function(1)`: prints `0`, calls `function(-1)`.
- `function(-1)` returns and `function(1)` prints 2.
- `function(3)` prints 4.
- `function(5)` prints 6.

The complete output sequence is:
`4 2 0 2 4 6`
So the correct option is **B**.

Q4 **Text Solution:**

- Loop x from `0` to 4:
  - When `x = 0`, `j = 1 + 0 = 1`
  - When `x = 1`, `j = 1 + 1 = 2`
  - When `x = 2`, `j = 2 + 2 = 4`
  - When `x = 3`, `j = 4 + 3 = 7`
  - When `x = 4`, `j = 7 + 4 = 11`
- `g(4)` returns 11.
- `k = count + g(x + 1) = 4 + 11 = 15`
- Update `count = count + x = 4 + 3 = 7`

Q5 **Text Solution:**
Step-by-step Analysis:

1. Initial Call: `f(a)`
   - `a = [11, 23, 34, 45, 56]`
   - `i[-1] = i[0]` modifies a to `[11, 23, 34, 45, 11]` (replaces the last element with the first element).
   - `j = i[:-2]` takes all elements except the last two: `j = [11, 23, 34]`.
   - `print(j)` prints `[11, 23, 34]`.
   - Next, `f(j[3:])` is called. Since `j = [11, 23, 34]`, `j[3:]` is an empty list `[]`.
2. Second Call: `f([])`
   - `i = []`
   - `len(i) == 0`, so the function returns immediately without any output.

Summary of Output:

- From the first call, the output is `[11, 23, 34]`.
- The second call does not produce any output.

So, the correct option is:
B

Q6 **Text Solution:**

1. **Calculate `fun(1)`**:
   - Since `n == 1`, the function returns 1.
2. **Calculate `fun(2)`**:
   - For `n = 2`, initialize `x = 1`.
   - The `for` loop runs with k in `range(1, 2)`, i.e., `k = 1`.

- Compute `x += fun(1) * fun(2 - 1)`, which is `fun(1) * fun(1)`.
- `fun(1) = 1`, so `x += 1 * 1 = 2`.
- Thus, `fun(2)` returns 2.

3. **Calculate `fun(3)`:**

- For `n = 3`, initialize `x = 1`.
- The `for` loop runs with k in `range(1, 3)`, i.e., `k = 1` and `k = 2`.
  - **For `k = 1`:**
    - Compute `x += fun(1) * fun(3 - 1)`, which is `fun(1) * fun(2)`.
    - `fun(1) = 1` and `fun(2) = 2`, so `x += 1 * 2 = 3`.
  - **For `k = 2`:**
    - Compute `x += fun(2) * fun(3 - 2)`, which is `fun(2) * fun(1)`.
    - `fun(2) = 2` and `fun(1) = 1`, so `x += 2 * 1 = 5`.
- Therefore, `fun(3)` returns 5.

In summary:

- `fun(1)` returns 1
- `fun(2)` returns 2
- `fun(3)` returns 5

Thus, the return value of `fun(3)` is **5**.

Q7 **Text Solution:**

Let's trace the function call `foo(513, 2)`:

1. **First Call: `foo(513, 2)`**

- Since `513 > 0`, compute `(513 % 2) + foo(513 // 2, 2)`.
- `513 % 2` is 1.
- `513 // 2` is 256.
- So, the call becomes `1 + foo(256, 2)`.

2. **Second Call: `foo(256, 2)`**

- Since `256 > 0`, compute `(256 % 2) + foo(256 // 2, 2)`.
- `256 % 2` is 0.
- `256 // 2` is 128.
- So, the call becomes `0 + foo(128, 2)`.

3. **Third Call: `foo(128, 2)`**

- Since `128 > 0`, compute `(128 % 2) + foo(128 // 2, 2)`.
- `128 % 2` is 0.
- `128 // 2` is 64.
- So, the call becomes `0 + foo(64, 2)`.

4. **Fourth Call: `foo(64, 2)`**

- Since `64 > 0`, compute `(64 % 2) + foo(64 // 2, 2)`.
- `64 % 2` is 0.
- `64 // 2` is 32.
- So, the call becomes `0 + foo(32, 2)`.

5. **Fifth Call: `foo(32, 2)`**

- Since `32 > 0`, compute `(32 % 2) + foo(32 // 2, 2)`.
- `32 % 2` is 0.
- `32 // 2` is 16.
- So, the call becomes `0 + foo(16, 2)`.

6. **Sixth Call: `foo(16, 2)`**

- Since `16 > 0`, compute `(16 % 2) + foo(16 // 2, 2)`.
- `16 % 2` is 0.
- `16 // 2` is 8.
- So, the call becomes `0 + foo(8, 2)`.

7. **Seventh Call: `foo(8, 2)`**

- Since `8 > 0`, compute `(8 % 2) + foo(8 // 2, 2)`.
- `8 % 2` is 0.
- `8 // 2` is 4.
- So, the call becomes `0 + foo(4, 2)`.

8. **Eighth Call: `foo(4, 2)`**

- Since `4 > 0`, compute `(4 % 2) + foo(4 // 2, 2)`.
- `4 % 2` is 0.
- `4 // 2` is 2.
- So, the call becomes `0 + foo(2, 2)`.

9. **Ninth Call: `foo(2, 2)`**

- Since `2 > 0`, compute `(2 % 2) + foo(2 // 2, 2)`.
- `2 % 2` is `0`.
- `2 // 2` is `1`.
- So, the call becomes `0 + foo(1, 2)`.

10. **Tenth Call: `foo(1, 2)`**
    - Since `1 > 0`, compute `(1 % 2) + foo(1 // 2, 2)`.
    - `1 % 2` is `1`.
    - `1 // 2` is `0`.
    - So, the call becomes `1 + foo(0, 2)`.
11. **Eleventh Call: `foo(0, 2)`**
    - Since `0 <= 0`, the function returns `0`.

Combining all these, the calculations are:

- `foo(1, 2)` returns `1 + 0 = 1`.
- `foo(2, 2)` returns `0 + 1 = 1`.
- `foo(4, 2)` returns `0 + 1 = 1`.
- `foo(8, 2)` returns `0 + 1 = 1`.
- `foo(16, 2)` returns `0 + 1 = 1`.
- `foo(32, 2)` returns `0 + 1 = 1`.
- `foo(64, 2)` returns `0 + 1 = 1`.
- `foo(128, 2)` returns `0 + 1 = 1`.
- `foo(256, 2)` returns `0 + 1 = 1`.
- `foo(513, 2)` returns `1 + 1 = 2`.

Thus, the output printed by `print(foo(513, 2))` is:

Q8 **Text Solution:**

Tracing `fun(3)`

1. **Call `fun(3)`**
   - Since `3 > 0`, x is updated to 2.
   - Call `fun(2)`.
2. **Call `fun(2)`**
   - Since `2 > 0`, x is updated to 1.
   - Call `fun(1)`.
3. **Call `fun(1)`**
   - Since `1 > 0`, x is updated to `0`.
   - Call `fun(0)`.
4. **Call `fun(0)`**
   - Since `0 <= 0`, the function returns immediately without printing anything.

   Now, back to **`fun(1)`**:
   - Print x, which is `0`.
   - Call `fun(0)`.
5. **Call `fun(0)`**
   - Again, the function returns immediately without printing anything.

   Back to **`fun(1)`**:
   - After the second call to `fun(0)`, x is updated to `-1`, but this does not affect the output.

   The output from **`fun(1)`** is `0`.
   Back to **`fun(2)`**:
   - Print x, which is `1`.
   - Call `fun(1)`.
6. **Call `fun(1)`**
   - This call has already been fully executed above, so it will print `0` as analyzed before.

   Back to **`fun(2)`**:
   - The output from **`fun(1)`** is `0`.
   - After the second call to `fun(1)`, x is updated to `-1`, but this does not affect the output.

   The output from **`fun(2)`** is `101`.
   Back to **`fun(3)`**:
   - Print x, which is `2`.
   - Call `fun(2)`.

   We have already computed the output of **`fun(2)`** as `101`, so the output from **`fun(2)`** is `101`.
   Finally, the output from **`fun(3)`** is `2012010`.

Combining the Output
Putting it all together, the output is:
2012010
Thus, the correct option is **B**.