## EXPRESSION EVALUATION [15 marks]

<table>
<thead>
<tr>
<th>Question 1 [1 mark].</th>
<th>Question 2 [1 mark].</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>print(max(-1, 2, -3))</code></td>
<td><code>print(1 + 6 // 4)</code></td>
</tr>
<tr>
<td><strong>Answer:</strong> (B) 2</td>
<td><strong>Answer:</strong> (A) 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Question 3 [1 mark].</th>
<th>Question 4 [1 mark].</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>print('abcde'[-1:7])</code></td>
<td><code>print(bool('False'))</code></td>
</tr>
<tr>
<td><strong>Answer:</strong> (B) e</td>
<td><strong>Answer:</strong> (A) True</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Question 5 [1 mark].</th>
<th>Question 6 [1 mark].</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>print('' in 'abc')</code></td>
<td><code>print([1, 3] * 2)</code></td>
</tr>
<tr>
<td><strong>Answer:</strong> (A) True</td>
<td><strong>Answer:</strong> (C) [1, 3, 1, 3]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Question 7 [1 mark].</th>
<th>Question 8 [1 mark].</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>print((1, [2], 3)[1])</code></td>
<td><code>print([[1, 2], [3, 4, 5]][1][2])</code></td>
</tr>
<tr>
<td><strong>Answer:</strong> (B) [2]</td>
<td><strong>Answer:</strong> (A) 5</td>
</tr>
</tbody>
</table>
**Question 9** [1 mark].

```python
print(sorted('abracadabra')[:3])
```

*Answer: (B) ['a', 'a', 'a']*

**Question 10** [1 mark].

```python
print({1: 2, 2: 1}[1] + {3: 4, 0: 1}[0])
```

*Answer: (A) 3*

**Question 11** [1 mark].

```python
print({1: 2, 3: 4}.get(2))
```

*Answer: (D) None*

**Question 12** [1 mark].

```python
print({1: {2: {3: 4}}}[{1: 2, 3: 4}[1]])
```

*Answer: (E) Evaluating this expression yields an error*

**Question 13** [1 mark].

```python
print([i - 1 for i in [1, 2]])
```

*Answer: (A) [0, 1]*

**Question 14** [1 mark].

```python
print([i for i in [0, 1, 2] if i - 1])
```

*Answer: (A) [0, 2]*

**Question 15** [1 mark].

```python
a = map(int, '123')
print(max(a) + min(a))
```

*Answer: (E) Evaluating this expression yields an error*

---

**MULTIPLE STATEMENT QUESTIONS** [15 marks]

**Question 16** [3 marks]. Which of the following statements is true of lists and tuples?

*Answer: (E) None of the above*
Question 17 [3 marks]. Observe the following code snippet and some remarks about it:

```
1 a = int(input())
2 cond = a == 1
3 if cond == True:
4     print(1)
5 else:
6     print('not 1')
```

Answer: (E) (1) Line 3 can be replaced with `if cond:` and the code snippet would behave identically; (3) Upon execution of this snippet, a `ValueError` will be raised from line 1 if the user enters a floating point number into the console.

Question 18 [3 marks]. Observe the following memoized implementation of the fibonacci function:

```
memo = {}
def fib(n):
    if n == 0 or n == 1:
        return 1
    if n in memo:
        return memo[n]
    z = fib(n - 1) + fib(n - 2)
    memo[n] = z
    return z
```

Which of the following statements is true of `fib`?

Answer: (A) Memoizing `fib` using a list is just as (if not more) efficient than memoizing it using a dictionary.

Question 19 [3 marks]. Observe the `Duck` class:

```
class Duck:
    def __init__(self):
        self.sound = 'quack'
```

Which of the following statements is true of `Duck`?

Answer: (C) The expression `Duck().sound` will always evaluate to `'quack'`.

Question 20 [3 marks]. Which of the following is true of exceptions?

Answer: (A) Raising exceptions can be useful for detecting errors.
PROGRAM TRACING [20 marks]

Question 21 [4 marks].

```python
def f21(n):
    if n > 5: return 5
    if n > 3: return 3
    if n > 1: return 1
    return 0

print(f21(2))
```

Answer: (B) 1

Question 22 [4 marks].

```python
def f22(seq):
    if isinstance(seq, str):
        return seq
    return ''.join([f22(i) for i in seq])

print(f22(['a', ['b', ['c']], [['d']]]))
```

Answer: (C) abcd

Question 23 [4 marks].

```python
def f23(d):
    acc = {}
    for k, v in d.items():
        if v not in acc:
            acc[v] = []
            acc[v].append(k)
    return acc

print(f23({1: 2, 2: 3, 3: 2, 4: 2, 5: 4}))
```

Answer: (C) {2: [1, 3, 4], 3: [2], 4: [5]}
Question 24 [4 marks].

```python
f = lambda y: lambda x: x[-y]
ls = [['i', 'am'], ['an', 'SoC'], [], ['student']]
_ = map(f, range(1, 5))
_ = map(lambda f: f(ls), _)
_ = map(' '.join, _)
res = ' '.join(_)
print(res)
```

Answer: (A) student an SoC i am

Question 25 [4 marks].

```python
class Entity:
    def reset(self):
        self.uuid = 0

class Named(Entity):
    def reset(self):
        self.name = ''
        super().reset()

class WithEmail(Entity):
    def reset(self):
        self.email = ''
        super().reset()

class User(Named, WithEmail):
    def __init__(self):
        self.name = 'Bob'
        self.email = 'bob@gmail.com'
        self.uuid = 123
    def reset(self):
        super().reset()

bob = User()
bob.reset()
print([bob.uuid, bob.name, bob.email])
```

Answer: (C) [0, '', '']
**PROGRAMMING** [50 marks]

**Question 26** [5 marks]. *Answer:*

```python
def cheapest(A, x, y):
    a = sum(A[min(x, y):max(x, y) + 1]) - A[x]
    b = sum(A[:min(x, y) + 1] + A[max(x, y):]) - A[x]
    return min(a, b)
```

**Question 27** [8 marks]. *Answer:*

```python
def deep_dup(seq):
    if seq == []:
        return seq
    if isinstance(seq[0], list):
        return [deep_dup(seq[0])] + deep_dup(seq[1:])
    return [seq[0]] * 2 + deep_dup(seq[1:])
```

**Question 28 (i)** [5 marks]. *Answer:*

```python
def weighted_sum_i(num_str, weight):
    output = 0
    for i in range(len(num_str)):
        output += int(num_str[i]) * weight[i]
    return output
```

**Question 28 (ii)** [5 marks]. *Answer:*

```python
def weighted_sum_r(num_str, weight):
    if not num_str:
        return 0
    return int(num_str[0]) * weight[0] + weighted_sum_r(num_str[1:], weight[1:])
```

**Question 28 (iii)** [5 marks]. *Answer:*

```python
def weighted_sum_1(num_str, weight):
    return sum(int(num_str[i]) * weight[i] for i in range(len(num_str)))
```

**Question 29** [6 marks]. *Answer:*

```python
def create_guess_game(n):
    return lambda x: 'bingo' if x == n else '
    'too big' if x > n else '
    'too small'
```
**Question 30** [8 marks]. Answer:

```python
def all_descendants(name, dd):
    output = [name]
    if name not in dd:
        return output
    for des in dd[name]:
        output += all_descendants(des, dd)
    return output
```

**Question 31 (i)** [4 marks]. Answer:

```python
def sum_2D(m, r_start, r_end, c_start, c_end):
    output = 0
    for i in range(r_start, r_end):
        for j in range(c_start, c_end):
            output += m[i][j]
    return output
```

**Question 31 (ii)** [4 marks]. Answer:

```python
def sum_2D(m, r_start, r_end, c_start, c_end):
    return sum(m[i][j] for i in range(r_start, r_end)
               for j in range(c_start, c_end))
```

– End of Solutions Manual –