Indian Statistical Institute
BSDS Semester-I 2026-2027
Mid-semester Examination (30 September, 2026)
Subject: Introduction to Computing
Total: 25 marks Duration: 1 hr.

Solutions + grading guideline for PART A

1. Suppose the following statements are executed one by one at the Python prompt. For each line containing a call to `print()`, state what output will be generated, or whether an error message will be printed. You do not need to copy the statements; just the statement number and the output will suffice. Provide a very brief justification (1-2 lines) for each answer. [17]

```
1  s = "3.1415926e2"
2  print(int(s))
3  print(int(float(s)))
4  print(bool(None))
5  print(bool(-100))
6  print(bool(-0))
7  x = 2
8  print("Exam is for " + x + " hours")
9  print("Exam is for ", x, " hours")
10 print(x * "BSDS")
11 print("x" * "BSDS")
```

```
12 l = ["x", "y", "z", "xy", "yz", "zx", "xyz"]
13 print(sorted(l))
14 l = [2, 10, 20, 1]
15 print(sorted(l))
16 print(l)
17 l.sort()
18 print(l)
19 print(sorted(["1", "2", "10", "20"]))
20 y = 3
21 x, y = y, x+y
22 print(x, y)
```

Answer:

2. 1 mark ERROR; `e` cannot occur in an integer.
3. 1 mark 314; `float(s)` returns `314.15926`, which is then truncated, i.e., the fractional part is just dropped.
4. 1 mark `False` (no explanation required).
5. 0.5 mark `True`.
6. 0.5 mark `False`.

   For (5) and (6) together: any non-zero value evaluates to `True`, zero evalutates to `False` → + 0.5 mark for this explanation.

8. 1 mark ERROR; cannot concatenate int to str.
9. 0.5 mark `Exam is for 2 hours` (no explanation required).
10. 1 mark `BSDSBSDS`; concatenates `BSDS` twice.
11. 0.5 mark ERROR; cannot multiply two strings.
13. 2 marks `['x', 'xy', 'xyz', 'y', 'yz', 'z', 'zx']`; strings are sorted in alphabetical / lexical order.
15. 1 mark `[1, 2, 10, 20]`; numbers are sorted in numerical order.
16. 1 mark `[2, 10, 20, 1]`; `sorted()` returns a sorted copy, does not change the actual list.
17. 0.5 mark no output (no explanation required). Full credit if no answer is provided, but deduct 0.5 marks if students write anything as output.

18. 1 mark `[1, 2, 10, 20]`; `sort()` sorts the list in place.

19. 1 mark `['1', '10', '2', '20']`; strings are sorted in alphabetical / lexical order.

22. 1 mark for `x`, 2 marks for `y` `3 5`; during `x, y = y, x+y`, current value of `y` (i.e., 3) is assigned to `x`; current values of `x` (2) and `y` (3) are used to calculate `x+y`, which is assigned to `y`.

- If the answer is incorrect, no marks should be given.
- If the answer is correct, and the student has clearly done rough work (either next to the question, or on the last page), but has not provided a formal explanation, full marks should be given.
- If the answer is correct, but an **incorrect** explanation is provided, give the student **50% credit** for the question.
- If the answer is correct, but no explanation is provided, give the student **70% credit** for the question.
- Different answers given by the same student may belong to different categories. Award fractional marks for each answer, based on its category, and round the total to the next multiple of 0.5. (Please let me know if this is unclear.)

2. (a) Without using the `itertools` library, write a function `cartesian_product(L1, L2)` that takes two lists `L1` and `L2` as input, and prints all pairs (X, Y) where X is an element of `L1`, and Y is an element of `L2`.

```
def cartesian_product(L1, L2):
    for i in L1:
        for j in L2:
            print("(" + str(i) + ", " + str(j) + ")")
            # Plain print(i, j) is also acceptable
```

OR

```
def cartesian_product(L1, L2):
    L = [ (i, j) for i in L1 for j in L2 ]
    for x in L:
        print(x)
```

If a student's answer is like the second alternative, but simply uses print(L), deduct 0.5 mark (since this will print extra [and]).

(b) If `L1` is `["a", 5.9, (1,2)]` and `L2` is `[[3.14, 159], "hi", (7,7)]`, what output will `cartesian_product(L1, L2)` produce? [2 + 1 = 3]

Just ensure that the output for part (b) is consistent with the `print()` calls used in part (a). If the content is correct, but the format is not what the student's solution to part (a) would produce, deduct 0.5 mark.

3. The following function `reverse_in_place(L)` is supposed to reverse a list `L` *in place*, without using the `reverse()` or `reversed` library functions. Fill in the blanks so that it works as desired. You do not need to copy the function; just write the part number, and what you would write in the corresponding blank. Very briefly explain your reasoning (5 lines or less). [5]

```
def reverse_in_place(L):
    for i in ____(a)____:
        L[ ____(b)____ ], L[ ____(c)____ ] = L[ ____(d)____ ], L[ ____(e)____ ]
    print(L)
    return L
```

(a) `range(len(L) // 2)` [2]

(b) `L[i]` [0.5]

(c) `L[-i-1]` or `L[len(L)-i-1]` [1]

(d) `L[-i-1]` or `L[len(L)-i-1]` [1]

(e) `L[i]` [0.5]

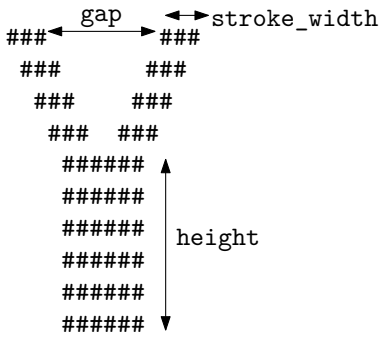
Indian Statistical Institute
BSDS Semester-I 2026-2027
Mid-semester Examination (30 September, 2026)
Subject: Introduction to Computing

Total: 30 marks **Maximum marks: 25** **Duration: 1 hr.**

PART B: This part of the exam is an open-book, open-notes exam.

You do not need to copy any code provided in the question paper. Just write the code for the missing parts, along with the corresponding part number, and a brief justification for your answer.

1. The figure below shows the letter Y drawn on the terminal using the `#` character.



An outline of a function `print_y_pattern` is given below. Fill in the blanks in the body of the function so that it draws a figure on the terminal (like the one shown above), using the values of the 3 parameters (dimensions) shown in the figure. One line and one column of the terminal occupy unit length in the vertical and horizontal direction, respectively; thus, in the above example, `stroke_width` is 3, `gap` is 8, and `height` is 6 (you may assume that `gap` is an even number). [8]

```
def print_y_pattern(stroke_width, gap, height):
    for i in ____(a)____:   range(gap//2 + 1)  or  range(gap//2)  or  range(gap,0,-2)
        print(' ' * ____(b)____ +  i  or  ((gap-i)//2)
              '#' * ____(c)____ +   stroke_width
              ' ' * ____(d)____ +  (gap-2*i)  or  i
              '#' * ____(e)____)   stroke_width
    for i in ____(f)____:   range(height-1)   or   range(height)
        print(' ' * ____(g)____ +   (gap//2)
              '#' * ____(h)____)   (2*stroke_width)
```

- Marks breakup: for (c) and (e), 0.5 mark each; for (g), 2 marks; for all other blanks, 1 mark.
- Deduct 0.5 marks for small mistakes e.g.,
  - use of division (/) instead of integer division (//),
  - student's answer is different from desired answer by 1.

2. *Hangman* is a game played by two players. One of the players A thinks of a word (say w), consisting of m letters, and presents it to the other player B, with some/all letters missing. B tries to guess the word, one letter at a time. If the letter guessed by B occurs in w, A fills in **all** the positions in w where the letter occurs; otherwise, B loses a 'life'. The process continues, until B correctly guesses the word, or runs out of lives.

   An outline of a function `hangman(word, guesses, lives)` is given below. It takes two strings and an integer as parameters (the parameters are supposed to have self-explanatory names), displays the progress of the game, and finally prints whether

   - B has correctly guessed the word (in this case, if there are any extra letters in `guesses`, they are ignored);
   - B has run out of lives; or
   - the game is incomplete.

   Fill in the blanks in the body of the function so that it works as intended. For example, if the function is called as `hangman("lion", "euiaoblnxy", 6)`, it should print the following. [10]

```
_ _ _ _      6 lives remaining
_ _ _ _      "e" guessed, 5 lives remaining
_ _ _ _      "u" guessed, 4 lives remaining
_ i _ _      "i" guessed, 4 lives remaining
_ i _ _      "a" guessed, 3 lives remaining
_ i o _      "o" guessed, 3 lives remaining
_ i o _      "b" guessed, 2 lives remaining
l i o _      "l" guessed, 2 lives remaining
l i o n      "n" guessed, 2 lives remaining
Word correctly guessed.
```

```
def hangman(word, guesses, lives):
    remaining_letters = set(____(a)____)   word
    current_status = ['_' for i in ____(b)____]   word
    for c in guesses:
        if c in remaining_letters:
            ____(c)____    remaining_letters.remove(c)
            for i, l in enumerate(word):
                if ____(d)____:   l == c:
                    ____(e)____    current_status[i] = c
        else:
            ____(f)____    lives -= 1

        print(''.join(current_status), '\t', c, 'guessed,', lives, 'lives remaining')
        if ____(g)____:    not remaining_letters
            print('Word correctly guessed')
            return
        elif ____(h)____:    lives == 0:
            print('Game over')
            return
    print('Game incomplete')
    return
```

(a) `word`; other longer equivalents, e.g., `[c for c in word]` also acceptable without penalty.

(b) `word`; other longer equivalents, e.g., `range(len(word))` also acceptable without penalty.

(c) `remaining_letters.remove(c)` or `remaining_letters.discard(c)`

(d) `l == c`

(e) `current_status[i] = c`

(f) `lives -= 1`

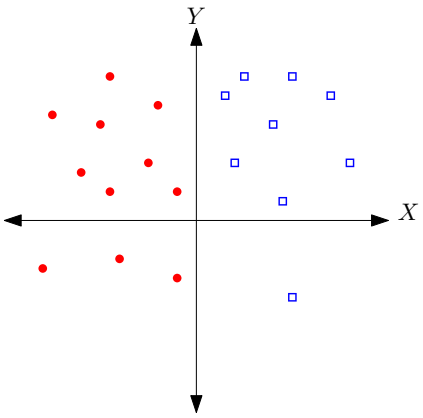
(g) `not remaining_letters` or `len(remaining_letters) == 0` or `''.join(current_status) == word`

(h) `lives == 0`

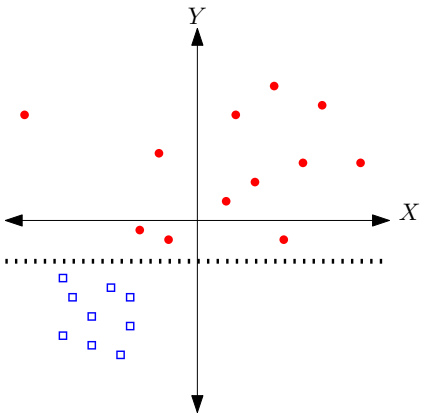
1.25 marks for each part; round to next multiple of 0.5.

Instead of filling in the blanks, if a student writes their own different solution, then s/he gets 6/10 for a correct answer. Marks are deducted proportionately for mistakes.

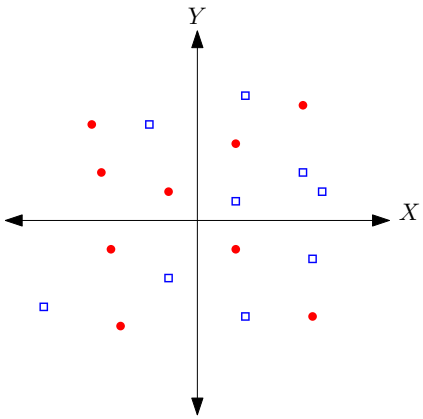
3. You are given the coordinates of some points in the X-Y plane. $N_R > 0$ of the points are coloured red (denoted by dots in the figures below), while the remaining $N_B > 0$ points are coloured blue (denoted by squares in the figures below). The following program is supposed to determine whether the red and blue points can be separated by a horizontal and/or vertical line. Fill in the blanks so that the program works as intended. [12]



The red and blue points can be separated by the Y-axis (and infinitely many other lines parallel to the Y-axis).



The red and blue points can be separated by the dotted line, which is parallel to the X-axis.



The red and blue points cannot be separated by any line that is parallel to the X or the Y-axis.

Input format: If there are N points, the input data will consist of N lines, each of which will contain 3 space-separated fields: 2 floating point numbers corresponding to the X and Y coordinates of the point, and a single letter, `R` or `B`, denoting the colour of the point.

```
min_red_x = max_red_x = min_red_y = max_red_y = None
min_blue_x = max_blue_x = min_blue_y = max_blue_y = None
while True:
  s = input()
  if s == "": # To end the input, the user hits ENTER without typing any data
    break
  x, y, colour = ____(1)____
  x = ____(2)____
  y = ____(3)____
  match colour:
    case 'R':
      if min_red_x == None:
        ____(4)____
      min_red_x = ____(5)____
      max_red_x = ____(6)____
      min_red_y = ____(7)____
      max_red_y = ____(8)____
```

```
    case 'B':
      if min_blue_x == None:
        _____(9)_____
      min_blue_x = ____(10)____
      max_blue_x = ____(11)____
      min_blue_y = ____(12)____
      max_blue_y = ____(13)____
if ___________(14)___________:
  print("Points can be separated by a line parallel to the Y axis")

if ___________(15)___________:
  print("Points can be separated by a line parallel to the X axis")
```

(1) `s.split()` [1]

(2) `float(x)` [1]

(3) `float(y)` [0.5]

(4) `min_red_x, max_red_x, min_red_y, max_red_y = x, x, y, y` [1.5]

(5) `min(min_red_x, x)` [0.5]

(6) `max(max_red_x, x)` [0.5]

(7) `min(min_red_y, y)` [0.5]

(8) `max(max_red_y, y)` [0.5]

(9) `min_blue_x, max_blue_x, min_blue_y, max_blue_y = x, x, y, y` [1]

(10) `min(min_blue_x, x)` [0.5]

(11) `max(max_blue_x, x)` [0.5]

(12) `min(min_blue_y, y)` [0.5]

(13) `max(max_blue_y, y)` [0.5]

(14) `max_red_x < min_blue_x or max_blue_x < min_red_x` [1.5]

(15) `max_red_y < min_blue_y or max_blue_y < min_red_y` [1.5]

For (14) and (15):
1 out of 4 conditions omitted $\Rightarrow$ 2.5/3.0,
2 out of 4 conditions omitted $\Rightarrow$ 1.5/3.0,
3 out of 4 conditions omitted $\Rightarrow$ 1.0/3.0.