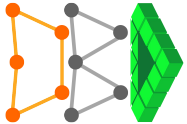
databased

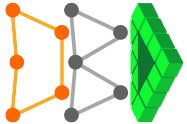
IISc's Computer Science Crew

BUG-A-THON

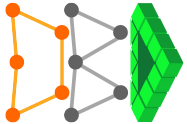
13 August, 2023



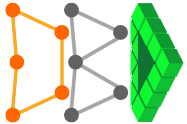
A SHORT STORY

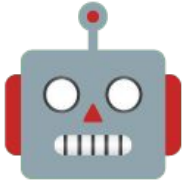


YOU HAPPY

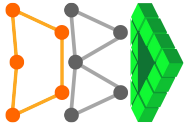


BECAUSE YOU HAVE GAME

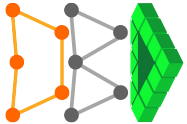




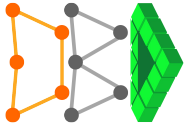
BUT VIRUS ATTACC



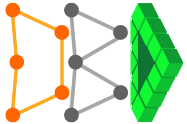
YOU SAD

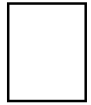


FRIEND WHO CAN CODE

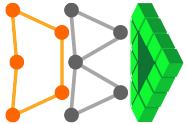


BUT BAD CODE

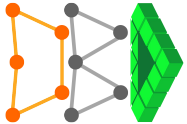




LEFT COMMENTS THOUGH

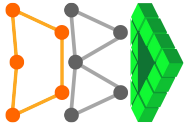


YOU DEBUG



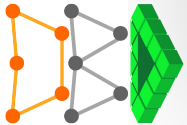
INSTRUCTIONS

1. Find out bugs in the code and explain in plain English on the forms.
2. Write as many bugs as you can find.
3. Each team should only submit once.
4. Feel free to ask us for clarifications!



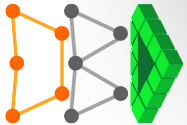
LENGTH (t.ly/eQkAH)

```
def length(s):
    """The function is calculates the number of element in s"""
    count = 0

    # ch iteratively takes on each character in s
    for ch in s:

        # Count only if ch is not an empty character
        if ch != "":
            count += 1

        # Count other characters
        count += 1

    return count
```



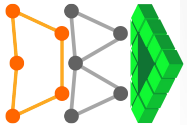
LENGTH

```
def length(s):
    """The function is calculates the number of element in s"""
    count = 0

    # ch iteratively takes on each character in s
    for ch in s:
        count += 1

    return count
```



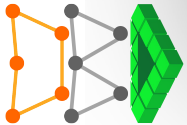
RANDOM RANGE (t.ly/82K8P)

```python
def random_range(a, b):
    """
    Return a random integer in the range [a, b).
    """
    # random.random() returns a random float in the range [0.0, 1.0)
    r = random.random()

    target = r * (b + a) - a

    n = 0
    while n < target:
        n += 1

    return n
```



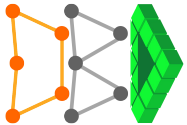
RANDOM RANGE

```
def random_range(a, b):
    """
    Return a random integer in the range [a, b).
    """
    # random.random() returns a random float in the range [0.0, 1.0)
    r = random.random()

    target = r * (b - a) + a

    n = 0
    while n <= target:
        n += 1

    return n-1
```



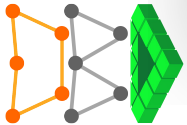
LINES (t.ly/0QQc_)

```
def lines(s):
    """
    Split the string s into a list of lines. A line ends with a newline character ('\n').
    """
    list_of_lines = [] # empty list

    # first and last are the indices of the first and last characters of the line being processed
    first = 0
    last = 0

    # i iteratively takes on each index in s
    i = 0
    while i < len(s):

        if s[i] == '\n':
            last = i - 1

            # s[first:last] returns a substring of s from index 'first' to index 'last' (not including
'last')
            # .append() adds the item to the end of the list
            list_of_lines.append(s[first:last])
            first = last + 1
            last = first + 1

        i += 1

    return list_of_lines
```



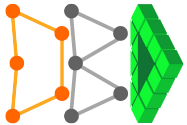
LINES

```
def lines(s):
    """
    Split the string s into a list of lines. A line ends with a newline character ('\n').
    """
    list_of_lines = []

    # first and last are the indices of the first and last characters of the line being processed
    first = 0
    last = 0

    # i iteratively takes on each index in s
    i = 0
    while i < len(s):

        if s[i] == '\n':
            last = i

            # s[first:last] returns a substring of s from index 'first' to index 'last' (not including
'last')
            # .append() adds the item to the end of the list
            list_of_lines.append(s[first:last])
            first = last + 1
            last = first

        i += 1

    if last < len(s) - 1:
        list_of_lines.append(s[last + 1:len(s)])

    return list_of_lines
```



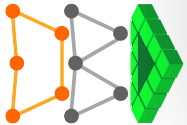
LOWER (t.ly/ShVmj)

```
def lower(s):
    """
    Return a copy of the string s with all characters lowercase.
    """
    new_string = ''

    # i iteratively takes on each index in s
    i = 0
    while i < len(s):
        ch = s[i]

        # ord(ch) returns the ASCII value of the character ch
        if ch > 'A' and ch < 'Z':
            # chr(n) returns the character with ASCII value n
            new_string += chr(ord(ch) + 32) # Upper case are 32 ASCII values behind lower case

        i += 1

    return new_string
```



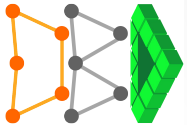
UPPER (t.ly/PDwn-)

```
def upper(s):
    """
    Return a copy of the string s with all characters uppercase.
    """
    new_string = ''

    # i iteratively takes on each index in s
    i = 0
    while i < len(s):
        ch = s[i]

        # ord(ch) returns the ASCII value of the character ch
        if ch > 'a' and ch < 'z':
            # chr(n) returns the character with ASCII value n
            new_string += chr(ord(ch) - 32) # Upper case are 32 ASCII values behind lower case

        i += 1

    return new_string
```



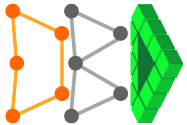
LOWER

```
def lower(s):
    """
    Return a copy of the string s with all characters lowercase.
    """
    new_string = ''

    # i iteratively takes on each index in s
    i = 0
    while i < len(s):
        ch = s[i]

        # ord(ch) returns the ASCII value of the character ch
        if ch >= 'A' and ch <= 'Z':
            # chr(n) returns the character with ASCII value n
            new_string += chr(ord(ch) + 32) # Upper case are 32 ASCII values behind lower case
        else:
            new_string += ch
        i += 1

    return new_string
```



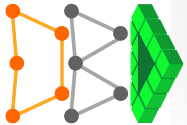
UPPER

```
def upper(s):
    """
    Return a copy of the string s with all characters uppercase.
    """
    new_string = ''

    # i iteratively takes on each index in s
    i = 0
    while i < len(s):
        ch = s[i]

        # ord(ch) returns the ASCII value of the character ch
        if ch >= 'a' and ch <= 'z':
            # chr(n) returns the character with ASCII value n
            new_string += chr(ord(ch) - 32) # Upper case are 32 ASCII values behind lower case
        else:
            new_string += ch
        i += 1

    return new_string
```



BUBBLE SORT (t.ly/dOsvH)

```
def sort(L):
    """
    returns the iterable Sorted list of alphabetical letters in ascending order.
    """
    # i iteratively takes on each index in L
    i = 0
    while i < len(L):

        # j iteratively takes on each index in L after i
        j = i + 2
        while j < len(L):

            # L[i] is the i-th element of L
            if L[i] < L[j]:

                # Swapping
                L[i] = L[j]
                temp = L[i]
                L[j] = temp

            j += 1
        i += 1

    return L
```

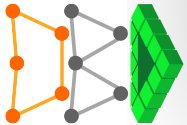


BUBBLE SORT

```
def sort(L):
    """
    returns the iterable Sorted list of alphabetical letters in ascending order.
    """
    # i iteratively takes on each index in L
    i = 0
    while i < len(L):

        # j iteratively takes on each index in L after i
        j = i + 1
        while j < len(L):

            # L[i] is the i-th element of L
            if L[i] > L[j]:
                temp = L[i]
                L[i] = L[j]
                L[j] = temp

            j += 1
        i += 1

    return L
```

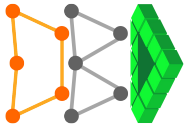
MERGE SORT (t.ly/epeDw)

```
def merge_sort(L):
    """
    Merge sort implementation
    """
    if len(L) <= 0:
        return L
    else:
        # // is floor division (divide and take the floor)
        mid = len(L) // 2

        # L[a:b] returns a sublist of L from index a to index b (not including b)
        left = merge_sort(L[0:mid-1])
        right = merge_sort(L[mid+1:len(L)-1])

        # merge the two sorted sublists
        return merge(left, right)
```



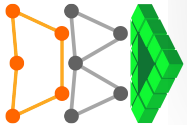
MERGE SORT

```
def sort2(L):
    """
    Merge sort implementation
    """
    if len(L) <= 1:
        return L
    else:
        # // is floor division (divide and take the floor)
        mid = len(L) // 2

        left = sort2(L[0:mid])
        right = sort2(L[mid:len(L)])

        # merge the two sorted sublists
        return merge(left, right)
```



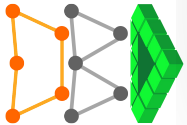
MERGE (t.ly/sBP0O)

```python
def merge(left, right):
    """
    Merge two sorted lists into a single sorted list.
    """
    result = []
    i = 0
    j = 0
    # left and right are sorted lists
    while i <= len(left) and j <= len(right):

        if left[i] >= right[j]:
            result.append(left[i]) # .append() adds the item to the end of the list
            i += 1
        else:
            result.append(right[j])

    # Add any remaining elements
    while j < len(right):
        result.append(right[j])
        j += 1

    return result
```



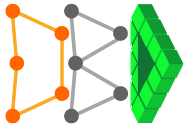
MERGE

```python
def merge(left, right):
    """
    Merge two sorted lists into a single sorted list.
    """
    result = []
    i = 0
    j = 0
    # left and right are sorted lists
    while i < len(left) and j < len(right):

        if left[i] <= right[j]:
            result.append(left[i]) # .append() adds the item to the end of the list
            i += 1
        else:
            result.append(right[j])
            j += 1

    while i < len(left):
        result.append(left[i])
        i += 1

    while j < len(right):
        result.append(right[j])
        j += 1

    return result
```



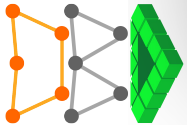
ORD (t.ly/rkful)

```
def ord(ch):
    """
    Return the ASCII value of the character ch, by performing binary search on the ASCII list.
    """
    # low and high are the indices of the first and last characters of the line being processed
    low = 0
    high = 127
    while low <= high:

        # math.ceil() returns the smallest integer greater than or equal to the input (Ceiling)
        mid = math.ceil((low + high) / 2)

        if ch == ASCII[mid]:
            return mid
        elif ch < ASCII[mid]:
            high = mid + 1
        else:
            low = mid - 1

    return -1
```



ORD

```
def ord(ch):
    """
    Return the ASCII value of the character ch, by performing binary search on the ASCII list.
    """
    # low and high are the indices of the first and last characters of the substring being searched
    low = 0
    high = 127
    while low <= high:

        # // is floor division (divide and take the floor)
        mid = (low + high) // 2

        if ch == ASCII[mid]:
            return mid
        elif ch < ASCII[mid]:
            high = mid - 1
        else:
            low = mid + 1

    return -1
```




THANK YOU
RESULTS ON TUESDAY

WHAT NEXT?

ANOTHER CONTEST!

DETAILS SOON