```python
# String-related functions without using inbuilt functions
def string_length(s):
    count = 0
    for _ in s:
        count += 1
    return count

def count_char(s, target):
    count = 0
    for ch in s:
        if ch == target:
            count += 1
    return count

def reverse_string(s):
    reversed_s = ''
    for i in range(string_length(s) - 1, -1, -1):
        reversed_s += s[i]
    return reversed_s

def is_palindrome(s):
    n = string_length(s)
    for i in range(n // 2):
        if s[i] != s[n - i - 1]:
            return False
    return True

def to_lower_case(s):
    result = ''
    for ch in s:
        if 'A' <= ch <= 'Z':
            result += chr(ord(ch) + 32)
        else:
            result += ch
    return result

def to_upper_case(s):
    result = ''
    for ch in s:
        if 'a' <= ch <= 'z':
            result += chr(ord(ch) - 32)
        else:
            result += ch
    return result

def remove_spaces(s):
    result = ''
    for ch in s:
        if ch != ' ':
            result += ch
    return result

def count_words(s):
    count = 0
    in_word = False
    for ch in s:
        if ch != ' ' and not in_word:
            count += 1
            in_word = True
        elif ch == ' ':
            in_word = False
    return count
```

```
def replace_char(s, old, new):
    result = ''
    for ch in s:
        if ch == old:
            result += new
        else:
            result += ch
    return result

def find_substring(s, sub):
    n = string_length(s)
    m = string_length(sub)
    for i in range(n - m + 1):
        match = True
        for j in range(m):
            if s[i + j] != sub[j]:
                match = False
                break
        if match:
            return i
    return -1

def remove_vowels(s):
    result = ''
    for ch in s:
        if ch not in 'aeiouAEIOU':
            result += ch
    return result

def get_unique_characters(s):
    result = ''
    for ch in s:
        exists = False
        for r in result:
            if ch == r:
                exists = True
                break
        if not exists:
            result += ch
    return result

def char_frequency(s):
    freq = {}
    for ch in s:
        if ch in freq:
            freq[ch] += 1
        else:
            freq[ch] = 1
    return freq

def capitalize_first_letter(s):
    result = ''
    capitalize = True
    for ch in s:
        if ch == ' ':
            result += ch
            capitalize = True
        elif capitalize and 'a' <= ch <= 'z':
            result += chr(ord(ch) - 32)
            capitalize = False
        else:
            result += ch
            capitalize = False
    return result
```

```python
# List-related functions without using inbuilt functions

def list_length(lst):
    count = 0
    for _ in lst:
        count += 1
    return count

def list_sum(lst):
    total = 0
    for num in lst:
        total += num
    return total

def list_max(lst):
    max_val = lst[0]
    for num in lst:
        if num > max_val:
            max_val = num
    return max_val

def list_min(lst):
    min_val = lst[0]
    for num in lst:
        if num < min_val:
            min_val = num
    return min_val

def count_occurrences(lst, target):
    count = 0
    for item in lst:
        if item == target:
            count += 1
    return count

def reverse_list(lst):
    reversed_lst = []
    for i in range(list_length(lst) - 1, -1, -1):
        reversed_lst.append(lst[i])
    return reversed_lst
def is_palindrome(lst):
    n = list_length(lst)
    for i in range(n // 2):
        if lst[i] != lst[n - i - 1]:
            return False
    return True

def remove_duplicates(lst):
    unique = []
    for item in lst:
        found = False
        for u in unique:
            if u == item:
                found = True
                break
        if not found:
            unique.append(item)
    return unique

def sort_list(lst):
    n = list_length(lst)
    for i in range(n):
        for j in range(i + 1, n):
            if lst[i] > lst[j]:
                lst[i], lst[j] = lst[j], lst[i]
    return lst
```

```
def merge_lists(lst1, lst2):
    result = []
    for item in lst1:
        result.append(item)
    for item in lst2:
        result.append(item)
    return result

def second_max(lst):
    max1 = lst[0]
    for num in lst:
        if num > max1:
            max1 = num
    max2 = None
    for num in lst:
        if num != max1:
            if max2 is None or num > max2:
                max2 = num
    return max2

def common_elements(lst1, lst2):
    common = []
    for i in lst1:
        for j in lst2:
            if i == j:
                already_added = False
                for k in common:
                    if k == i:
                        already_added = True
                if not already_added:
                    common.append(i)
    return common

def rotate_left(lst, k=2):
    n = list_length(lst)
    rotated = []
    for i in range(k, n):
        rotated.append(lst[i])
    for i in range(k):
        rotated.append(lst[i])
    return rotated

def split_even_odd(lst):
    even = []
    odd = []
    for num in lst:
        if num % 2 == 0:
            even.append(num)
        else:
            odd.append(num)
    return even, odd

def are_equal(lst1, lst2):
    if list_length(lst1) != list_length(lst2):
        return False
    for i in range(list_length(lst1)):
        if lst1[i] != lst2[i]:
            return False
    return True

def list_to_string(lst):
    result = ''
    for ch in lst:
        result += ch
    return result
```

```
def string_to_list(s):
    result = []
    for ch in s:
        result.append(ch)
    return result

def remove_duplicate_characters(s):
    result = ''
    for ch in s:
        found = False
        for r in result:
            if ch == r:
                found = True
                break
        if not found:
            result += ch
    return result

def string_length(s):
    count = 0
    for _ in s:
        count += 1
    return count

def alternate_case(s):
    result = ''
    upper = True
    for ch in s:
        if 'a' <= ch <= 'z' or 'A' <= ch <= 'Z':
            if upper:
                if 'a' <= ch <= 'z':
                    result += chr(ord(ch) - 32)
                else:
                    result += ch
                upper = False
            else:
                if 'A' <= ch <= 'Z':
                    result += chr(ord(ch) + 32)
                else:
                    result += ch
                upper = True
        else:
            result += ch
    return result

def reverse_each_word(s):
    result = ''
    word = ''
    for ch in s + ' ':
        if ch != ' ':
            word += ch
        else:
            for i in range(string_length(word)-1, -1, -1):
                result += word[i]
            result += ' '
            word = ''
    return result.strip()


# Tuple-related functions without using inbuilt functions
def tuple_length(t):
    count = 0
    for _ in t:
        count += 1
    return count

def count_occurrences(t, target):
    count = 0
    for item in t:
```

```
            if item == target:
                count += 1
        return count

    def find_max(t):
        max_val = t[0]
        for item in t:
            if item > max_val:
                max_val = item
        return max_val

    def find_min(t):
        min_val = t[0]
        for item in t:
            if item < min_val:
                min_val = item
        return min_val
    def tuple_sum(t):
        total = 0
        for item in t:
            total += item
        return total

    def index_of_element(t, value):
        index = 0
        for item in t:
            if item == value:
                return index
            index += 1
        return -1
    def reverse_tuple(t):
        result = ()
        for i in range(tuple_length(t) - 1, -1, -1):
            result += (t[i],)
        return result

    def convert_list_to_tuple(lst):
        result = ()
        for item in lst:
            result += (item,)
        return result

    def convert_tuple_to_list(t):
        result = []
        for item in t:
            result.append(item)
        return result

    def are_tuples_equal(t1, t2):
        if tuple_length(t1) != tuple_length(t2):
            return False
        for i in range(tuple_length(t1)):
            if t1[i] != t2[i]:
                return False
        return True

    # Dictionary-related functions without using inbuilt functions
    def dict_length(d):
        count = 0
        for _ in d:
            count += 1
        return count

    def get_keys(d):
        keys = []
        for key in d:
            keys.append(key)
        return keys

```

```
  def get_values(d):
      values = []
      for key in d:
          values.append(d[key])
      return values

  def get_items(d):
      items = []
      for key in d:
          items.append((key, d[key]))
      return items

  def key_exists(d, target):
      for key in d:
          if key == target:
              return True
      return False

  def value_exists(d, target):
      for key in d:
          if d[key] == target:
              return True
      return False

  def count_value_occurrences(d, target):
      count = 0
      for key in d:
          if d[key] == target:
              count += 1
      return count

  def add_key_value(d, key, value):
      d[key] = value
      return d

  def update_value(d, key, value):
      if key_exists(d, key):
          d[key] = value
      return d

  def delete_key(d, target):
      new_dict = {}
      for key in d:
          if key != target:
              new_dict[key] = d[key]
      return new_dict

  def merge_dicts(d1, d2):
      merged = {}
      for key in d1:
          merged[key] = d1[key]
      for key in d2:
          merged[key] = d2[key]
      return merged

  def dict_from_list(pairs):
      d = {}
      for pair in pairs:
          key, value = pair
          d[key] = value
      return d

  def invert_dict(d):
      inverted = {}
      for key in d:
          value = d[key]
          inverted[value] = key
      return inverted

```

```
484
485 def dict_get(d, key, default=None):
486     for k in d:
487         if k == key:
488             return d[k]
489     return default
490
491 def dict_has_key(d, key):
492     for k in d:
493         if k == key:
494             return True
495     return False
496
497 def dict_update(d1, d2):
498     for key in d2:
499         d1[key] = d2[key]
500     return d1
501
502 def dict_copy(d):
503     new_d = {}
504     for key in d:
505         new_d[key] = d[key]
506     return new_d
507
508 # Set-related functions without using inbuilt functions
509 def set_length(s):
510     count = 0
511     for _ in s:
512         count += 1
513     return count
514
515 def add_to_set(s, value):
516     exists = False
517     for item in s:
518         if item == value:
519             exists = True
520             break
521     if not exists:
522         s.add(value)
523     return s
524
525 def remove_from_set(s, value):
526     new_set = set()
527     for item in s:
528         if item != value:
529             new_set.add(item)
530     return new_set
531
532 def is_member(s, value):
533     for item in s:
534         if item == value:
535             return True
536     return False
537
538 def union_sets(s1, s2):
539     result = set()
540     for item in s1:
541         result.add(item)
542     for item in s2:
543         result.add(item)
544     return result
545
546 def intersection_sets(s1, s2):
547     result = set()
548     for item in s1:
549         for other in s2:
550             if item == other:
551                 result.add(item)
552     return result
```

```


def difference_sets(s1, s2):
    result = set()
    for item in s1:
        found = False
        for other in s2:
            if item == other:
                found = True
                break
        if not found:
            result.add(item)
    return result

def is_subset(s1, s2):
    for item in s1:
        found = False
        for other in s2:
            if item == other:
                found = True
                break
        if not found:
            return False
    return True

def convert_list_to_set(lst):
    result = set()
    for item in lst:
        result = add_to_set(result, item)
    return result

def convert_set_to_list(s):
    result = []
    for item in s:
        result.append(item)
    return result

```