Assignment 10

1. a. Define a function that determines whether a given number is even or odd.

Solution:

```python
def is_even(n):
    return n % 2 == 0
```

b. Write a program that counts the number of even and odd numbers in a list.

Solution:

```python
def count_evens_odds(numbers):
    even_count = 0
    odd_count = 0
    for num in numbers:
        if is_even(num):
            even_count += 1
        else:
            odd_count += 1
    return even_count, odd_count
```

2. a. Explain the difference between a stack and a queue in terms of their operations and data structures.

Solution:

A stack is a Last-In-First-Out (LIFO) data structure, where elements are added to the top and removed from the top. A queue is a First-In-First-Out (FIFO) data structure, where elements are added to the back and removed from the front.

b. Implement a stack using a list in Python.

Solution:

```python
class Stack:
    def __init__(self):
        self.items = []

    def push(self, item):
        self.items.append(item)

    def pop(self):
        if not self.is_empty():
            return self.items.pop()
        else:
            return None

    def is_empty(self):
        return len(self.items) == 0
```

3. a. Calculate the factorial of a given number.

Solution:

```python
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)
```

b. Use the factorial function to compute the number of ways to arrange 5 items.

Solution:

```python
arrangements = factorial(5)
```

4. a. Implement a binary search algorithm in Python.

Solution:

```python
def binary_search(arr, target):
    low = 0
    high = len(arr) - 1
    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            low = mid + 1
        else:
            high = mid - 1
    return -1
```

b. Use the binary search function to find the index of a target value in a sorted list.

Solution:

```python
index = binary_search(sorted_list, target_value)
```

5. a. Write a program that generates all possible permutations of a given string.

Solution:

```python
from itertools import permutations

def generate_permutations(s):
    return [''.join(p) for p in permutations(s)]
```

b. Use the permutation function to list all possible arrangements of the letters in the word "example".

Solution:

```python
permutations = generate_permutations("example")
```

6. a. Implement a function that calculates the area of a circle given its radius.

Solution:

```python
import math

def circle_area(radius):
    return math.pi * radius**2
```

b. Use the area function to find the area of a circle with a given radius.

Solution:

```python
area = circle_area(radius)
```