Assignment-3

Part-A

Pseudocode

```
SmartPriorityQueue(Generic[K]):
    constructor(capacity: int, comparison_function: Callable[[K, K], bool]):
        heap_array: List[K] = []
        size = 0
        is_min_heap = True
        comparator = comparison_function

    insert( element: K):
        heapq.heappush(heap_array, element)
        size += 1

    top() -> K:
        if is_empty():
            raise ValueError
        return heap_array[0]

    toggle():
        is_min_heap = not is_min_heap
        if is_min_heap:
            comparator = lambda a, b: a < b
        else:
            comparator = lambda a, b: a > b
        heapq.heapify(heap_array)

    remove_top() -> K:
        if is_empty():
            raise ValueError
        return heapq.heappop(heap_array)

    remove(element: K) -> K:
        if is_empty():
```

```
        raise ValueError
    heap_array.remove(element)
    heapq.heapify(heap_array)
    return element

replace_key( old_key: K, new_key: K) -> K:
    if is_empty():
        raise ValueError
    remove(old_key)
    insert(new_key)
    return old_key

replace_value( old_value: K, new_value: K) -> K:
    if is_empty():
        raise ValueError
    index = heap_array.index(old_value)
    heap_array[index] = new_value
    heapq.heapify(heap_array)
    return old_value

status() -> str:
    return "Min" if is_min_heap else "Max"

is_empty() -> bool:
    return size == 0

size() -> int:
    return size

to_array() -> List[K]:
    return heap_array
```

Part - B

Tight Big-O time complexities

- **insert: O(log n)**
- **top: O(1)**
- **toggle: O(n)**
- **remove_top: O(log n)**
- **remove: O(n)**
- **replace_key: O(n)**
- **replace_value: O(n)**
- **status, is_empty, size: O(1)**