Date: 19.1.2024

Session Topic: Subsets and Subsequences

Task 1

Question: Minimum Size Subarray sum

Solution:

```
class Solution {

    public int minSubArrayLen(int target, int[] nums) {

        int left=0,sum=0,min=Integer.MAX_VALUE;

        for(int right=0;right<nums.length;right++)

        {

            sum+=nums[right];

            while(sum>=target)

            {

                sum-=nums[left];

                min=Math.min(min,right-left+1);

                left++;

            }

        }

        if(min==Integer.MAX_VALUE)

        {

            return 0;

        }

        return min;

    }

}
```

Date: 19.1.2024

Session Topic: Subsets and Subsequences

Task 2

Question: Maximum Subarray

Solution:

```
class Solution {

    public int maxSubArray(int[] nums) {

        int curr=0;

        int max=Integer.MIN_VALUE;

        for(int i=0;i<nums.length;i++)

        {

            curr+=nums[i];

            if(curr<nums[i])

            {

                curr=nums[i];

            }

            else

            {

                curr=curr;

            }

            if(curr>max)

            {

                max=curr;

            }

        }

        return max;
```

```
    }

}
```

Date: 19.1.2024

Session Topic: Subsets and Subsequences

Task 3

Question: Find Subsequence of Length K With the Largest Sum

Solution:

```
class Solution {

    public int[] maxSubsequence(int[] nums, int k) {

        int arr[]=new int[k];

        PriorityQueue<Integer>pq=new PriorityQueue<>();

        List<Integer>list=new ArrayList<>();

        for(int i=0;i<nums.length;i++)

        {

            pq.add(nums[i]);

            list.add(nums[i]);

        }

        System.out.println(pq);

        for(int i=0;i<nums.length-k;i++)

        {

            list.remove(pq.remove());

        }

        System.out.println(pq);

        for(int i=0;i<arr.length;i++)

        {
```

```
            arr[i]=list.get(i);

        }

        return arr;

    }

}
```

Date: 19.1.2024

Session Topic: Subsets and Subsequences

Task 4

Question: Longest Increasing Subsequence

Solution:

```
class Solution {

    public int lengthOfLIS(int[] nums) {

        int temp[]=new int[nums.length];

        int max=1;

        temp[0]=nums[0];

        for(int i=1;i<nums.length;i++)

        {

            if(temp[max-1]<nums[i])

            {

                temp[max++]=nums[i];

            }

            else

            {

                int index=Arrays.binarySearch(temp,0,max,nums[i]);

                if(index<0)
```

```
                {

                    index=-(index+1);

                }

                temp[index]=nums[i];

            }



        }

        //System.out.print(Arrays.toString(temp));

        return max;

    }

}
```

Date: 19.1.2024

Session Topic: Subsets and Subsequences

Task 5

Question: Longest Increasing Subsequence

Solution: Subarray sums equals K

```
class Solution {

    public int subarraySum(int[] nums, int k) {

        int count=0;

        int arr[]=new int[nums.length+1];

        arr[0]=0;

        int index=1;

        for(int i=0;i<nums.length;i++)

        {

            arr[index++]=arr[i]+nums[i];
```

```
        }
        for(int i=0;i<arr.length;i++)
        {
            for(int j=i+1;j<arr.length;j++)
            {
                if(arr[j]-arr[i]==k)
                {
                    count++;
                }
            }
        }
        return count;
    }
}
```

Date: 19.1.2024

Session Topic: Subsets and Subsequences

Task 5

Question: Longest Increasing Subsequence

Solution: Contiguous Array

```
class Solution {
    public int findMaxLength(int[] nums) {
        int sum=0,max=0;
        Map<Integer,Integer>map=new HashMap<>();
        map.put(0,-1);
        for(int i=0;i<nums.length;i++)
```

```
        {
            sum+=nums[i]==0?-1:1;
            if(map.containsKey(sum))
            {
                max=Math.max(max,i-map.get(sum));
            }
            else
            {
                map.put(sum,i);
            }
        }
        return max;
    }
}
```