Experiment no 08

Code:

```
//Set Operations
console.log("Set Operations");
console.log("Union Operation");
function union(a,b){
    let unionSet=new Set(a);
    for(let i of b){
        unionSet.add(i);
    }
    return unionSet;
}
const setA=new Set(['apple','mango','orange']);
const setB=new Set(['grapes','apple','banana']);
const result=union(setA,setB);
console.log(result);

//InterSection Operation
console.log("InterSection Operation");
function interSection(setA,setB){
    let intersectionSet=new Set();
    for(let i of setB){
        if(setA.has(i)){
            intersectionSet.add(i);
        }
    }
    return intersectionSet;
}
const a=new Set(['apple','mango','orange']);
const b=new Set(['greps','apple','banana']);
const result_2=interSection(a,b);
console.log(result_2);

//difference Operation
console.log("Difference Operation");
function difference(setA,setB){
    let differenceSet=new Set(setA);
    for(let i of setB){
        differenceSet.delete(i);
    }
    return differenceSet;
}
const c=new Set(['apple','mango','orange']);
const d=new Set(['greps','apple','banana']);
const result_3=difference(c,d);
console.log(result_3);
```

```
//Subset Operation
console.log("SubSet Operation");
function subset(setA,setB){
    for(let i of setB){
        if(!setA.has(i)){
            return false;
        }
    }
    return true;
}

const e=new Set(['apple','mango','orange']);
const f=new Set(['apple','orange']);
const result_4=subset(e,f);
console.log(result_4);
```

Output:

```
PS D:\College_JavaScript_Practical> node
"d:\College_JavaScript_Practical\Experiment_8.js"
Set Operations
Union Operation
Set(5) { 'apple', 'mango', 'orange', 'grapes', 'banana' }
InterSection Operation
Set(1) { 'apple' }
Difference Operation
Set(2) { 'mango', 'orange' }
SubSet Operation
true
```