ES6 & TypeScript Assignments

1. **Promises:** Create 2 promises, one generates value of x & another generates value of you. Write a program to print sum of x & y. (Use Promise.all)

```
const sum = (a, b) => a + b;

const first = new Promise((resolve, reject) => {
  const x = 5;
  if (x) resolve(x);
  else reject(x);
});

const second = new Promise((resolve, reject) => {
  const y = 10;
  if (y) resolve(y);
  else reject(y);
});

const allpromises = Promise.all([first, second]);
allpromises.then(sucess => console.log('sum:', sum(sucess[0],
sucess[1])))
  .catch(error => console.log('error: ', error))
  .finally(() => console.log("Executed finally block"));
```

```
>> sum: 15
>> Executed finally block
```

2. **TypeScript classes & types:** Write a class Account with attributes id, name, balance. Add two sub classes SavingAccount & CurrentAccount having specific attribute interest & cash_credit respectively. Create multiple saving & current account objects. Write a functionality to find out total balance in the bank.

```
class Account {
  id: number;
  name: string;
  balance: number;

  // Normal signature with defaults
  constructor(id = 1, name = "abc", balance = 1000) {
    this.id = id;
    this.name = name;
    this.balance = balance;
  }
}

class SavingsAccount extends Account {
  interest: number;
  constructor(id, name, balance, interest) {
    super(id, name, balance);
    this.interest = interest;
  }
  totalBalance() {
    let newBalance = this.balance * this.interest;
    this.balance = this.balance + newBalance;
    return this.balance;
  }
}

class CurrentAccount extends Account {
  cash_credit: number;
  constructor(id, name, balance, cash_credit) {
    super(id, name, balance);
    this.cash_credit = cash_credit;
  }
  totalBalance() {
    let newBalance = this.balance * this.cash_credit;
    this.balance = this.balance + newBalance;
    return this.balance;
  }
}

let saving = new SavingsAccount("11110001", "abc", 2000, 1.5);
let current = new CurrentAccount("22220001", "xyz", 5000,
0.5);
console.log(saving.totalBalance());
console.log(current.totalBalance());
```

```
>> 5000
>> 7500
```

3. **TypeScript Interfaces:** Write an interface Printable. Create 2 objects circle & employee those implement Printable interface. Write a function printAll() that takes all objects as argument & invoke print() method on every object.

```
interface Printable {
  name:string,
  printAll: (string) => string
}

var circle:Printable = {
  name:"abc",
  printAll: (str) => {return "hi "+ str}
}

var employee:Printable = {
  name:"xyz",
  printAll: (str) => {return "hello "+ str}
}

console.log(circle.printAll(circle.name))
console.log(employee.printAll(employee.name))
```

```
>> hi abc
>> hello xyz
```