```
//Desmond Mpofu

//Question 1
// ● Count the number of times a char appears in a string
const countChars = (str, tofind)=>{
    let count =0;
    for(let i=0; i<str.length; i++){
        if(str.charAt(i)===tofind){
        count=count+1;
        }
    }
    return count
}
//Test string
console.log(countChars("This is the test sentence try it",
"t"))

//-------------------------------------------------------------
-------------

// ● Find whether or not a substring is present in a larger
string
const subStr = (str, sub) =>{
    if(str.indexOf(sub)===-1){
        return false
    }else{
        return true
    }
}
//Test string
console.log(subStr("This is the story of the substring",
"tha"))

//-------------------------------------------------------------
-------------

//● Sort an arbitrary array of integers in ascending order,
bonus marks for using one of
//the more efficient algorithms
```

```
const arrSort = (arr)=>{
    arr.sort((a,b)=>(a-b))

    console.log(arr)
}

arrSort([2,4,7,3,9,4,8,3,1])

//-------------------------------------------------------------
-------------

//Question 2: Two dimensional Collision detection
//SQUARES
//This code assumes the squares are sitting horizontal to x-
xis and areboth not rotated in any way
const detect =(sqr1_x, sqr1_y,sqr1_width, sqr2_x,
sqr2_y,sqr2_width)=>{


    if(sqr1_x >sqr2_x + sqr2_width||
      sqr1_x + sqr1_width<sqr2_x ||
      sqr1_y>sqr2_y+sqr1_width||
      sqr1_y+sqr1_width<sqr2_y){
          return "There is no collusion"
      }else{
          return "Big bang..."
      }
}
//Arbitrary x&y coordinates and width of two squares to test
console.log(detect(4, 6, 5, 5,8,6))

//-------------------------------------------------------------
-------------


//CIRCLES
const doTheyCollide =(circle1_x,circle1_y,radius1,
circle2_x,circle2_y,radius2,)=>{
```

```
    let dx= circle2_x - circle1_x;
    let dy =circle2_y-circle1_y;
    //pythagoras theorem to determine distance btwn 2 circles'
centers
    let distance = Math.sqrt( dx*dx + dy*dy);
    let radius = radius1+radius2;
    if(distance<radius){
        return "Points lie within other circle"
    }else{
        return "No collusion"
    }
}
//Arbitrary x&y coordinates and radius of two circles to test
console.log(doTheyCollide(2, 4, 2, 8,7, 3))


//Question 3: Packets in a queue

function PriorityQueue(){
  let items = [];

  function QueueElement(element, priority){
    this.element = element;
    this.priority = priority;
  }

  //Add a new element/packet in queue
  this.enqueue = function(element, priority){
    let queueElement = new QueueElement(element, priority);

    let added = false;
    for(let i = 0; i < items.length; i++){
      //We are using giving priority to higher numbers
      //If new element has more priority then add it at that
place
      if(queueElement.priority > items[i].priority){
        items.splice(i, 0, queueElement);

        added = true;
```

```
        break;
      }
    }

    //Then add it to the end of the queue
    if(!added){
      items.push(queueElement);
    }
  }

  //Remove element from the queue
  this.dequeue = () => {
    return items.shift();
  }

  //Return the first element from the queue
  this.front = () => {
    return items[0];
  }

  //Return the last element from the queue
  this.rear = () => {
    return items[items.length - 1];
  }


  this.isEmpty = () => {
    return items.length == 0;
  }

  this.size = () => {
   return items.length;
  }

  //Print the queue
  this.print = function(){
    for(let i = 0; i < items.length; i++){
      console.log(`${items[i].element} - ${items[i].priority}
`);
```

```
    }
  }
}

let pQ = new PriorityQueue();
pQ.enqueue(1, 3);
pQ.enqueue(5, 2);
pQ.enqueue(6, 1);
pQ.enqueue(11, 1);
pQ.enqueue(13, 1);
pQ.enqueue(10, 3);
pQ.dequeue();
pQ.print();
```