Q2:

1.Prototypal inheritance and classical inheritance are like doing same things using only different ways like everything in Javascript is objects and inheriting from them and that is the reason why javascript is called as oops based language and prototypal inheritance is done by adding methods or functions to prototype making it accessible everywhere in the program and we can use class to do the samething with different manner.

2.In javascript inheritance using prototypes is done for accessing the basic methods associated with the types of data-structure being worked with.while class based inheritance limits its usage only with classes.

3.The advantages are accessibility throughout the program and no need to declare again and again and can also be made non- accessible with static and # key word.and as well as disadvantage is creates some extra space .

4.for ex1:

Class Person{

Constructor(name,age,dept){

This.name=name;

This.age=age;

This.dept=dept;

}

getInfo(){

return `$ {this.name=name}

${this.age=age}

${this.dept=dept}`;

}

}

Here getInfo() method is prototypallly inheritable in whatever new class I create.

Likewise ex2:

Let getInfo={

Info(){

return `$ {this.name=name}

${this.age=age}

${this.dept=dept}`;

}

}

Function person(name,age,dept){

Let person={name,age,dept};

Person.\_\_proto\_\_=getInfo;

Return person;

}

Here in this getInfo is made or set into prototype to access anywhere like

Let person1=person(“someone”,25,”cs”);

Let person2=person(“someone”,25,”IT”);

console.log(person1.Info());

console.log(person2.Info());