Answer for Mid-Semester Questions (CSET104), Object oriented Programming Using Java

Q1.

Automatic conversion of primitive data types to their corresponding Wrapper class object is autoboxing. Automatic conversion of a Wrapper class object to its corresponding primitive type is auto autounboxing. **[1]**

Example: int i=2;
Integer I=i;**(autoboxing)**
int j=I**(auto-unboxing)** **[1]**

Byte b=2;
double d=b.doubleValue();
Double, Byte, Integer, Float are child classes of Abstract Number class and **doubleValue()** is defined in Number class. **[1]**

Q2.

S1==S2: Compares if S1 and S2 references of same Object.
S1.equals(S2): Returns true or false depending upon if S1 and S2 have same String values
S1.compareTo(S2): Returns +ve vale if S1 is lexicographically greater than S2, 0 if S1 and S2 are equal and -ve if S1 is lexicographically smaller than S2. **[1+1+1]**

Q3.

```
public static void printAlphabet(double o)
{   int a=(int)Math.floor(o);
    if ((a>='A' && a<='Z') ||(a>='a' && a<='z'))
    {
       System.out.println((char) a);
    }
    else
       System.out.println("Invalid Input");

}
```

[3]

Q4.

Error: Parent class has a parameterized constructor, however child class constructor is not calling it.
Change the Square class constructor as **[1]**

```
class Square extends Rectangle{
   Square(int x){super(x,x);}}
```

[2]

Or

Define a default constructor in Parent class

```
Rectangle(){}
```

Q5.

Yes, it is possible to overload a parent class method in child class. **[1]**

Example

```
class Parent{
   static void area(int a)
   {
      System.out.println("Area of a square:"+a*a);

   }
}
class Child
{
   static void area(int a, int b)
   {
      System.out.println("Area of a Rectangle:"+a*b);

   }
}
```
[2]

(i)In method overloading, though the name of the methods is same the signatures are different. In method Overriding the name and signature of the methods are same in parent and child classes.(ii) Method overriding involves parent and child relationship whereas in case of method overloading it can happen with a single class. (iii) Dynamic dispatch is available with method overriding not with overloading. **(Any two differences)** **[2]**

No, we cannot qualify an overridden method as protected in child class if it is qualified as public in parent class as we are attempting to assign weaker access privileges in child class. **[1]**

Q6.

```
class Student implements Comparable<Student>
{  static Scanner s=new Scanner(System.in);
   String name;
   double cgpa;
   int rank;
   Student()
   {
      name=s.nextLine();
      cgpa=s.nextDouble();
      s.nextLine(); //for newline character from previous entry
   }
   void setRank(int i)
   {
      rank=i;
   }
   public String toString()
   {
      return name+", Rank: "+rank;
   }
   public int compareTo(Student s)
   {
      if(cgpa==s.cgpa)
         return 0;
```

```
        else if(cgpa>s.cgpa)
            return 1;
        else
            return -1;

    }
}
public class Main {

    public static void main(String[] args)
    {
        ArrayList<Student> arr=new ArrayList<>();
        for(int i=0;i<5;i++)
            arr.add(new Student());
        Collections.sort(arr,Collections.reverseOrder());
        for(int i=0;i< arr.size();i++)
            arr.get(i).setRank(i+1);
        System.out.println(arr);
    }
}
```

Q7.

```
interface Time
{   int getHour();
    int getMinute();
    default public boolean equals(Time o)
    {   int H1=getHour();
        int M1=getMinute();
        int H2=o.getHour();
        int M2=o.getMinute();
        return (H1*60+M1)==(H2*60+M2);
    }
}
class Time1 implements Time
{ int h,m;
    Time1(int a, int b)
    {
        h=a;
        m=b;
    }
    public int getMinute()
    {
        return m;
    }
    public int getHour()
    {
        return h;
```

```
    }
}
class Time2 implements Time
{ int h,m,s;
    Time2(int a, int b, int c)
    {
        h=a;
        m=b;
        s=c;
    }
    public int getMinute()
    {
        return m;
    }
    public int getHour()
    {
        return h;
    }
}
```