**Matthew Clark**

**Assignment 8:**

**Group1:**

**What output is produced by the following code?**

**1.**

 1 public class inheritance1 {  
 2   public static void main (String[] args) {  
 3    Sub obj = new Sub();  
 4    obj.display();  
 5    }  
 6 }  
 7   
 8 class Super {  
 9   int num = 10;  
10   int id = 20;  
11 }  
12   
13 class Sub extends Super {  
14   int num = 100;  
15   public void display() {  
16      System.out.println(num);  
17      System.out.println(super.num);  
18      System.out.println(id);  
19   }  
20 }  
21

**Output:**

ΜΜ«Μ ----jGRASP exec: java inheritance1  
ΜΜ§Μ100  
ΜΜ§Μ10  
ΜΜ§Μ20  
ΜΜ§Μ  
ΜΜ©Μ ----jGRASP: operation complete.

**2.**

 1 class parent {  
 2   public void display() {  
 3     System.out.println("partent display(); ");  
 4   }  
 5 }  
 6   
 7 class child extends parent {  
 8   @Override  
 9   public void display () {  
10     System.out.println("child display();  ");  
11   }  
12   public void test() {  
13     display();  
14   }  
15 }  
16   
17 public class overriding1 {  
18   public static void main(String[] args) {  
19      child obj = new child();  
20      obj.display();  
21      obj.test();  
22   }  
23 }  
24

**Output:**

ΜΜ«Μ ----jGRASP exec: java overriding1  
ΜΜ§Μchild display();    
ΜΜ§Μchild display();    
ΜΜ§Μ  
ΜΜ©Μ ----jGRASP: operation complete.

**3.**

 1 class parent2 {  
 2   public void display() {  
 3     System.out.println("partent display(); ");  
 4   }  
 5 }  
 6   
 7 class child2 extends parent2 {  
 8   @Override  
 9   public void display() {  
10     super.display();  
11     System.out.println("child display();  ");  
12   }  
13   public void test() {  
14     display();  
15     super.display();  
16   }  
17 }  
18   
19 public class overriding2 {  
20   public static void main(String[] args) {  
21      child2 obj = new child2();  
22      obj.test();  
23      obj.display();  
24   }  
25 }  
26

**Output:**

ΜΜ«Μ ----jGRASP exec: java overriding2  
ΜΜ§Μpartent display();   
ΜΜ§Μchild display();    
ΜΜ§Μpartent display();   
ΜΜ§Μpartent display();   
ΜΜ§Μchild display();    
ΜΜ§Μ  
ΜΜ©Μ ----jGRASP: operation complete.

**4.**

 1 class superCon {  
 2   
 3    superCon(){  
 4      System.out.println("super constructor");  
 5    }  
 6  }  
 7    
 8  class subCon extends superCon{  
 9    subCon(){  
10      System.out.println("sub constructor");  
11    }  
12  }  
13    
14  public class constrctor1 {  
15   public static void main(String[] args) {  
16     subCon obj = new subCon();  
17  }  
18 }  
19

**Output:**

ΜΜ«Μ ----jGRASP exec: java constrctor1  
ΜΜ§Μsuper constructor  
ΜΜ§Μsub constructor  
ΜΜ§Μ  
ΜΜ©Μ ----jGRASP: operation complete.

**5.**

 1 class ParentClass2 {  
 2    public ParentClass2() {}    
 3    public void method1() {  
 4       System.out.println("ParentClass method1()"); }  
 5    public void method2() {  
 6       System.out.println("ParentClass method2()");}  
 7 }   
 8   
 9 class ChildClass2 extends ParentClass2 {  
10    public ChildClass2() { }  
11   
12    @Override  
13    public void method1() {  
14       System.out.println("ChildClass method1()"); }  
15      
16    public void method3(){  
17       System.out.println("ChildClass method3()"); }  
18 }  
19   
20 public class constructor2 {  
21    public static void main(String[] args) {  
22       ChildClass2 child = new ChildClass2();  
23       child.method1();  
24       child.method2();  
25       child.method3();}  
26 }  
27

**Output:**

ΜΜ«Μ ----jGRASP exec: java constructor2  
ΜΜ§ΜChildClass method1()  
ΜΜ§ΜParentClass method2()  
ΜΜ§ΜChildClass method3()  
ΜΜ§Μ  
ΜΜ©Μ ----jGRASP: operation complete.

**6.**

 1 class ParentClass3 {  
 2    public ParentClass3() {  
 3       System.out.println("Parent Class.");  }  
 4      
 5    public void method1() {  
 6       System.out.println("Parent Class : Method1()."); }  
 7 }  
 8   
 9 class ChildClass3 extends ParentClass3 {  
10    public ChildClass3() {  
11       System.out.println("Child Class.");}  
12      
13    @Override  
14    public void method1() {  
15       super.method1();  
16       System.out.println("Child Class : Method1().");}  
17 }  
18   
19 public class constructor3 {  
20    public static void main(String[] args) {  
21       ChildClass3 child = new ChildClass3();  
22       child.method1();}  
23 }  
24

**Output:**

ΜΜ«Μ ----jGRASP exec: java constructor3  
ΜΜ§ΜParent Class.  
ΜΜ§ΜChild Class.  
ΜΜ§ΜParent Class : Method1().  
ΜΜ§ΜChild Class : Method1().  
ΜΜ§Μ  
ΜΜ©Μ ----jGRASP: operation complete.

**Group 2:**

**1. What output is produced by the following code?**

 1 class Car{  
 2   String Color;  
 3   int door;  
 4   void drive(){  
 5     System.out.println( "drive, Car" ); }  
 6    
 7    void stop(){  
 8     System.out.println( "stop, Car" );}  
 9 }  
10   
11 class FireEngine extends Car {  
12   void drive(){  
13     System.out.println( "drive, FireEngine" );}  
14 }  
15 class Ambulance extends Car{  
16   void stop(){  
17     System.out.println( "stop, Ambulance" );}  
18 }  
19   
20 public class poly3{  
21   public static void main( String[] args ){  
22   Car c1 = new FireEngine();  
23   Car c2 = new Ambulance();    
24   c1.drive();  
25   c1.stop();  
26   c2.drive();  
27   c2.stop();  
28   }  
29 }  
30

**Output:**

ΜΜ«Μ ----jGRASP exec: java poly3  
ΜΜ§Μdrive, FireEngine  
ΜΜ§Μstop, Car  
ΜΜ§Μdrive, Car  
ΜΜ§Μstop, Ambulance  
ΜΜ§Μ  
ΜΜ©Μ ----jGRASP: operation complete.

**2. What output is produced by the following code?**

 1 class A{  
 2     int i = 10;  
 3     public String x(){return "A.x";}  
 4 }  
 5 class B extends A{  
 6     int i = 5;  
 7     public String x(){return "B.x";}  
 8     public String y(){return "y";}  
 9 }  
10 class B2 extends A{  
11     int i = 3;  
12     public String x(){return "B2.x";}  
13 }  
14 public class Poly4 {  
15     public static void main(String[] args) {  
16         A obj = new B();  
17         A obj2 = new B2();  
18         System.out.println(obj.x());  
19         System.out.println(obj2.x());  
20         System.out.println(obj.i);  
21         System.out.println(obj2.i);  
22     }  
23 }  
24

**Output:**

ΜΜ«Μ ----jGRASP exec: java Poly4  
ΜΜ§ΜB.x  
ΜΜ§ΜB2.x  
ΜΜ§Μ10  
ΜΜ§Μ10  
ΜΜ§Μ  
ΜΜ©Μ ----jGRASP: operation complete.

**3. Complete Java program to produce the following output.**

 1 class Vehicle {  
 2     String name;  
 3     String company;  
 4     int price;  
 5     public Vehicle(String name, String company, int price) {  
 6         this.name = name;  
 7         this.company = company;  
 8         this.price = price; }  
 9     public void printInfo() {  
10         System.out.println("car type: " + name + "Maker: " + company + "Price: " + price);  
11     }  
12 }  
13    
14 class Car extends Vehicle {  
15     int passengers;   
16     Car(String name, String company, int price, int passengers) {  
17         super(name, company, price);  
18         this.passengers = passengers; }  
19     @Override     
20         public void printInfo() {  
21             System.out.println("car type: " + name + " Maker: " + company + " Price: " + price + " Passengers: " + passengers);  
22     }  
23 }  
24    
25 class Truck extends Vehicle {  
26     int maxLoad;  
27     Truck(String name, String company, int price, int maxLoad) {  
28         super(name, company, price);  
29         this.maxLoad = maxLoad; }  
30     @Override     
31     public void printInfo() {  
32             System.out.println("car type: " + name + " Maker: " + company + " Price: " + price + " Max Load: " + maxLoad);  
33     }  
34 }  
35   
36 public class PolymorphismDemo {  
37     public static void main(String[] args) {    
38         Vehicle v1 = new Car("Sonata", "Hyundai", 3000, 5);  
39         Vehicle v2 = new Truck("Ranger", "Ford", 3500, 6);  
40         v1.printInfo();    
41         v2.printInfo();  }  
42 }  
43   
44 /\*output  
45 car type: Sonata  Maker: Hyundai    Price: 3000    Passengers: 5    
46 car type: Ranger  Maker: Ford    Price: 3500    Max Load: 6\*/

**Group 3:**

**1. Complete Java program to produce the following output.**

 1 interface Printable{    
 2    void print();    
 3 }    
 4   
 5 interface Showable{    
 6    void show();    
 7 }    
 8     
 9 class p3\_1 implements Printable,Showable{    
10    public void print() {  
11       System.out.println("Hello");  
12    }  
13    public void show() {  
14       System.out.println("Welcome");  
15    }  
16   
17    public static void main(String args[]){    
18       p3\_1 obj = new p3\_1();    
19       obj.print();    
20       obj.show();    
21    }    
22 }    
23 /\*output  
24 Hello  
25 Welcome\*/

**2. Complete Java program to produce the following output.**

 1 interface Printable{    
 2    void print();    
 3 }    
 4   
 5 class p3\_1 implements Printable{    
 6    public void print() {  
 7       System.out.println("Hello");  
 8    }  
 9    public static void main(String args[]){    
10       p3\_1 obj = new p3\_1();    
11       obj.print();    
12    }    
13 }    
14 /\*output  
15 Hello\*/

**3. What output is produced by the following code?**

 1 interface Printable{    
 2   void print();  }    
 3   
 4 interface Showable extends Printable{    
 5   void show();  }    
 6   
 7 class p3\_3 implements Showable{    
 8   public void print(){System.out.println("Hello");}    
 9   public void show(){System.out.println("Welcome");}    
10   public static void main(String args[]){    
11   p3\_3 obj = new p3\_3();    
12   obj.print();    
13   obj.show();  }    
14 }    
15

**Output:**

ΜΜ«Μ ----jGRASP exec: java p3\_3  
ΜΜ§ΜHello  
ΜΜ§ΜWelcome  
ΜΜ§Μ  
ΜΜ©Μ ----jGRASP: operation complete.

**4. Complete Java program to produce the following output.**

 1 // Example of Method Overloading  
 2 class p3\_4{    
 3    public void sum(double sum1, double sum2) {  
 4       System.out.println(sum1 + sum2);  
 5    }  
 6    public void sum(int sum1, int sum2) {  
 7       System.out.println(sum1 + sum2);  
 8    }  
 9    public static void main(String args[]){    
10       p3\_4 obj=new p3\_4();    
11       obj.sum(10.5,10.5);    
12       obj.sum(20,20);    
13    }    
14 }  
15 /\*output  
16 21.0  
17 40\*/

**5. What output is produced by the following code?**

 1 class Animal{    
 2 void eat(){System.out.println("animal is eating...");}    
 3 }    
 4     
 5 class Dog extends Animal{    
 6 void eat(){System.out.println("dog is eating...");}    
 7 }    
 8     
 9 class p3\_5 extends Dog{    
10 public static void main(String args[]){    
11 Animal a=new p3\_5();    
12 a.eat();    
13 }  
14 }  
15

**Output:**

ΜΜ«Μ ----jGRASP exec: java p3\_5  
ΜΜ§Μdog is eating...  
ΜΜ§Μ  
ΜΜ©Μ ----jGRASP: operation complete.

**6. What output is produced by the following code?**

 1 abstract class Bike{    
 2    Bike(){System.out.println("bike is created");}    
 3    abstract void run();    
 4    void changeGear(){System.out.println("gear changed");}    
 5  }    
 6     
 7  class Honda extends Bike{    
 8  void run(){System.out.println("running safely..");}    
 9  }    
10    
11  class p3\_6{    
12  public static void main(String args[]){    
13   Bike obj = new Honda();    
14   obj.run();    
15   obj.changeGear();    
16  }    
17 }    
18

**Output:**

ΜΜ«Μ ----jGRASP exec: java p3\_6  
ΜΜ§Μbike is created  
ΜΜ§Μrunning safely..  
ΜΜ§Μgear changed  
ΜΜ§Μ  
ΜΜ©Μ ----jGRASP: operation complete.

**Group 4:**

**1. Complete Java program to produce the following output.**

 1 import java.util.\*;  
 2 public class ex6\_1ItrRemoveElement {  
 3   
 4 public static void main(String[] args){  
 5    String removeElem = "apple";  
 6    ArrayList<String> myList = new ArrayList<String>();  
 7    myList.add("banana");  
 8    myList.add("apple");  
 9    myList.add("mango");  
10    myList.add("orange");  
11    myList.add("cherry");  
12    System.out.print("Before remove:");  
13    System.out.println(myList);  
14      
15    Iterator<String> itr = myList.iterator();  
16    while(itr.hasNext()){  
17       if(itr.next() == "apple"){  
18          itr.remove();  
19       }  
20     }  
21    System.out.print("After remove:");  
22    System.out.println(myList);  
23   }  
24 }  
25   
26 /\* ouput  
27 Before remove: [banana, apple, mango, orange, cherry]  
28 After remove: [banana, mango, orange, cherry]\*/

**2. Complete Java program to produce the following output.**

 1 import java.util.\*;  
 2 public class ex6\_2iterateHashMap {  
 3    public static void main(String[] args) {  
 4       HashMap< Integer, String> hMap = new HashMap<Integer, String>();  
 5       hMap.put(1, "Bellevue");  
 6       hMap.put(3, "College");  
 7       hMap.put(2, "CS211");  
 8       Collection cl = hMap.values();  
 9       Iterator itr = cl.iterator();  
10       while (itr.hasNext()) {  
11          System.out.println(itr.next());  
12       }  
13    }  
14 }  
15 /\*output  
16 Bellevue  
17 CS211  
18 College\*/

**3. What output is produced by the following code?**

 1  List list = Arrays.asList("one Two three Four five one three Four".split(" "));  
 2  System.out.println("List :"+list);  
 3  List sublist = Arrays.asList("three Four".split(" "));  
 4  System.out.println("SubList :"+sublist);  
 5  System.out.println("indexOfSubList: " + Collections.indexOfSubList(list, sublist));  
 6  System.out.println("lastIndexOfSubList: " + Collections.lastIndexOfSubList(list, sublist));  
 7            
 8  List sublist2 = Arrays.asList("Two Four".split(" "));  
 9  System.out.println("SubList :"+sublist2);  
10  System.out.println("indexOfSubList2: " + Collections.indexOfSubList(list, sublist2));

**Output:**

List :one Two three Four five one three Four

SubList :three Four

indexOfSubList: 3

lastIndexOfSublist: 7? Or 6?

SubList :Two Four

indexOfSubList2: -1

**4. What output is produced by the following code?**

 1  List list = Arrays.asList("1 2 3 4 5".split(" "));  
 2  System.out.println("List :"+list);  
 3  Collections.rotate(list, 2);  
 4  System.out.println("rotate: " + list);  
 5  Collections.rotate(list, 4);  
 6  System.out.println("rotate: " + list);  
 7   
 8  Collections.rotate(list, 3);  
 9  System.out.println("rotate: " + list);  
**Output:**

12345

45123

51234

23451

**5. What output is produced by the following code?**

 1  ArrayList<Integer> al = new ArrayList<Integer>();  
 2  HashSet<Integer> hs = new HashSet<Integer>();  
 3   for (int i= 0; i<5; i++) {  
 4      al.add(i);  
 5      hs.add(i);  
 6      hs.add(i\*2);  
 7   }      
 8  System.out.println("array");  
 9  Iterator ai = al.iterator();  
10     while(ai.hasNext()){  
11        System.out.println(ai.next());  
12     }  
13  Iterator hi = hs.iterator();  
14  System.out.println("hashset");  
15     while(hi.hasNext()){  
16       System.out.println(hi.next());  
17     }  
18

**Output:**

array

0

1

2

3

4

hashset

0

1

2

3

4

6

8

**6. Complete Java code to produce the following output.**

 1 import java.util.\*;  
 2   
 3 public class monsters {     
 4    public static void main(String[] args) {  
 5       // ArrayList   
 6       List a1 = new ArrayList();  
 7          a1.add("Seattle");  
 8          a1.add("Bellevue");  
 9          System.out.println("ArrayList: " + a1);  
10       // LinkedList  
11       LinkedList a2 = new LinkedList();  
12          a2.add("Seattle");  
13          a2.add("Bellevue");  
14          System.out.println("LinkedList: " + a2);  
15       // HashSet  
16       HashSet a3 = new HashSet();  
17          a3.add("Seattle");  
18          a3.add("Bellevue");  
19          System.out.println("HashSet: " + a3);  
20       // HashMap  
21       HashMap a4 = new HashMap();  
22          a4.put("Yakima", 2);  
23          a4.put("Seattle", 1);  
24          a4.put("Bellevue", 3);  
25          System.out.println("HashMap: " + a4);  
26    }       
27 }  
28 /\*output  
29 ArrayList: [Seattle, Bellevue]  
30 LinkedList: [Seattle, Bellevue]  
31 HashSet: [Seattle, Bellevue]  
32 HashMap: {Seattle=1, Yakima=2, Bellevue=3}\*/

**Group 5:**

**1. What output is produced by the following code?**

1 public class forLoopType2{  
2  public static void main (String[] args) {  
3    int myArray[] = {5,4,3,2,1,0};  
4    for (int num : myArray){  
5      System.out.println(num % 2);  
6    }  
7 } }  
8

**Output:**

ΜΜ«Μ ----jGRASP exec: java forLoopType2  
ΜΜ§Μ1  
ΜΜ§Μ0  
ΜΜ§Μ1  
ΜΜ§Μ0  
ΜΜ§Μ1  
ΜΜ§Μ0  
ΜΜ§Μ  
ΜΜ©Μ ----jGRASP: operation complete.

**2. What is the output of this program?**

 1 import java.util.\*;  
 2     class hashtableQ1 {  
 3         public static void main(String args[]) {  
 4             Hashtable obj = new Hashtable();  
 5             obj.put("A", new Integer(3));  
 6             obj.put("B", new Integer(2));  
 7             obj.put("C", new Integer(8));  
 8             System.out.print(obj.contains(new Integer(5)));  
 9         }  
10     }

**Output:**

ΜΜ«Μ ----jGRASP exec: java hashtableQ1  
ΜΜ§Μfalse  
ΜΜ©Μ ----jGRASP: operation complete.

**3. What is the output of this program?**

 1 import java.util.\*;  
 2     class hashtableQ1 {  
 3         public static void main(String args[]) {  
 4             Hashtable obj = new Hashtable();  
 5             obj.put("A", new Integer(3));  
 6             obj.put("B", new Integer(2));  
 7             obj.put("C", new Integer(8));  
 8             System.out.print(obj.contains(new Integer(5)));  
 9         }  
10     }

**Output:**

ΜΜ«Μ ----jGRASP exec: java hashtableQ1  
ΜΜ§Μfalse  
ΜΜ©Μ ----jGRASP: operation complete.

**4. What is the output of this program?**

 1 import java.util.\*;  
 2     class hashtableQ3 {  
 3         public static void main(String args[]) {  
 4             Hashtable obj = new Hashtable();  
 5             obj.put("A", new Integer(3));  
 6             obj.put("B", new Integer(2));  
 7             obj.put("C", new Integer(8));  
 8             obj.remove(new String("A"));  
 9             System.out.print(obj);  
10         }  
11     }  
12

**Output:**

ΜΜ«Μ ----jGRASP exec: java hashtableQ3  
ΜΜ§Μ{C=8, B=2}  
ΜΜ©Μ ----jGRASP: operation complete.

**5. What is the output of this program?**

 1 import java.util.\*;  
 2     class hashtableQ3 {  
 3         public static void main(String args[]) {  
 4             Hashtable obj = new Hashtable();  
 5             obj.put("A", new Integer(3));  
 6             obj.put("B", new Integer(2));  
 7             obj.put("C", new Integer(8));  
 8             obj.remove(new String("A"));  
 9             System.out.print(obj);  
10         }  
11     }  
12

**Output:**

ΜΜ«Μ ----jGRASP exec: java hashtableQ3  
ΜΜ§Μ{C=8, B=2}  
ΜΜ©Μ ----jGRASP: operation complete.

**Group 6:**

**1. Given the output stream A,B,C,D,E,F Write down the sequence of operations (Push for stack and Pop for unstack) which would produce the sequence C,B,D,E,F,A**

**2. What is the output of this program?**

 1 import java.util.\*;  
 2 public class LinkedListExample {  
 3      public static void main(String args[]) {  
 4          /\* Linked List Declaration \*/  
 5          LinkedList<String> linkedlist = new LinkedList<String>();  
 6          /\*add(String Element) is used for adding   
 7           \* the elements to the linked list\*/  
 8          linkedlist.add("Item1");  
 9          linkedlist.add("Item5");  
10          linkedlist.add("Item3");  
11          linkedlist.add("Item6");  
12          linkedlist.add("Item2");  
13          /\*Display Linked List Content\*/  
14          System.out.println("Linked List Content: " +linkedlist);  
15          /\*Add First and Last Element\*/  
16          linkedlist.addFirst("First Item");  
17          linkedlist.addLast("Last Item");  
18 System.out.println("LinkedList Content after addition: " +linkedlist);  
19 /\*This is how to get and set Values\*/  
20          Object firstvar = linkedlist.get(0);  
21          System.out.println("First element: " +firstvar);  
22 linkedlist.set(0, "Changed first item");  
23          Object firstvar2 = linkedlist.get(0);  
24          System.out.println("First element after update by set method: " +firstvar2);  
25          /\*Remove first and last element\*/  
26          linkedlist.removeFirst();  
27          linkedlist.removeLast();  
28          System.out.println("LinkedList after deletion of first and last element: " +linkedlist);  
29          /\* Add to a Position and remove from a position\*/  
30          linkedlist.add(0, "Newly added item");  
31          linkedlist.remove(2);  
32          System.out.println("Final Content: " +linkedlist);   
33      }  
34 }  
35

**Output:**

ΜΜ«Μ ----jGRASP exec: java LinkedListExample  
ΜΜ§ΜLinked List Content: [Item1, Item5, Item3, Item6, Item2]  
ΜΜ§ΜLinkedList Content after addition: [First Item, Item1, Item5, Item3, Item6, Item2, Last Item]  
ΜΜ§ΜFirst element: First Item  
ΜΜ§ΜFirst element after update by set method: Changed first item  
ΜΜ§ΜLinkedList after deletion of first and last element: [Item1, Item5, Item3, Item6, Item2]  
ΜΜ§ΜFinal Content: [Newly added item, Item1, Item3, Item6, Item2]  
ΜΜ§Μ  
ΜΜ©Μ ----jGRASP: operation complete.

**3. What is the running time of each the method?**

 1 public int add100(int[] array) {  
 2    if (array.length < 100) {  
 3       return 0;  
 4    }  
 5    int sum = 0;  
 6    for (int i = 0; i < 100; i++) {  
 7       sum += array[i];  
 8    }  
 9    return sum;  
10 }   
11   
12 public static void m4(int[] arr) {    
13    for (int i=0; i<arr.length; i++) {          
14       System.out.println(arr[i] \* 10);   
15    }     
16    for (int j=arr.length-1; j>=0; j--) {          
17       System.out.println(arr[j] / 10);   
18    }  
19 }  
20   
21 public static void m5(int[] arr) {     
22    for (int i=0; i<15; i++) {         
23       for (int j=0; j<arr.length; j++) {             
24          System.out.println(Math.max(arr[i],arr[j]));   
25       }  
26    }  
27 }

**4. Using big-O notation in terms of the parameter n, how much time does the following method take?**

 1 public static int mystery(int n) {  
 2     int count = 0;  
 3     int cur = 1;  
 4     while (cur < n) {  
 5         count++;  
 6         cur = cur \* 2;  
 7     }  
 8     return cur;  
 9 }

**5. Each of the below methods determine m, n without using Math.pow. Using big-O notation in terms of n, how much time does each take?**

 1 a.  
 2 int pow(int m, int n) {  
 3     int ret = 1;  
 4     for (int i = 0; i < n; i++) {  
 5         ret \*= m;  
 6     }  
 7     return ret;  
 8 }    
 9   
10 b.  
11 int pow(int m, int n) {  
12     int ret = 1;  
13     int k = m;  
14     int i = n;  
15     while (i > 0) {  
16         if (i % 2 == 1) ret \*= k;  
17         k \*= k;  
18         i /= 2;  
19     }  
20     return ret;  
21 }

**6. Why is binary search O(log2 N) ?**

**Group 7:**

**1. Create binary search tree and fill out the following array for {“apple”, “banana”, “cherry”, “date”, grape”, “mango”}**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Left Node | Data | Right Node |
| A[0] |  |  |  |
| A[1] |  |  |  |
| A[2] |  |  |  |
| A[3] |  |  |  |
| A[4] |  |  |  |
| A[5] |  |  |  |

**2. Create min heap tree for {7, 3, 5, 2, 4, 6}**

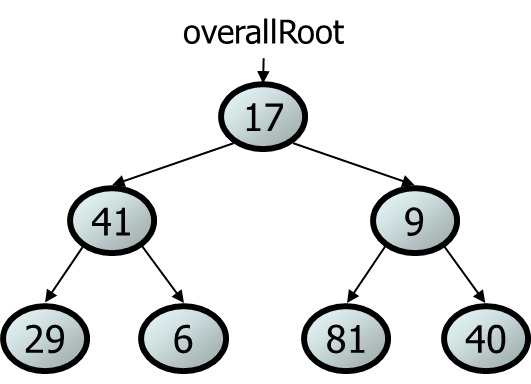
(2)

(3) (5)

(4)(7)(6)

**3. When the root is removed from the above heap tree, how do we alter the tree? Create a new heap tree step by step.**

**4. Find the following orderings for traversals.**



Pre-order (Root, Left, Right): 17, 41, 29, 6, 9, 81, 40

In-order (Left, Root, Right): 29, 41, 6, 17, 81, 9, 40

Post-order (Left, Right, Root): 29, 6, 41, 81, 40, 9, 17