السؤال 1

public class m1 {  
 class def {  
  
 def\_init(self,data):  
 self.data = data;  
 self.next = None;  
   
 node1 = Node(10);  
 node1 = Node(20);  
 node1 = Node(30);  
  
 node1.next = node2;  
 node1.next = node13;  
   
  
 }  
}

السؤال 2

public class m2 {  
  
 def\_init(self,data):  
 self.data = data;  
 self.next = None  
   
 class DoublylinkedList:  
 class def{  
 def is\_empty(self,data):  
 return self.head is None:  
   
 def prepend (self,data)  
 new\_node = Node(data)  
 if self.is\_empty():  
 else:  
 new\_node.prev=self.tail;  
 elf.tail.next=new\_node  
   
 def search(self,data):  
 while current.data is None:  
 if current.data ==  
 data  
   
 return current.next  
 return None;  
   
 def remove (self,data):  
 node = self.search (data)  
 if node.prev is None:  
 self.head = node.next  
 else:  
   
 if node.next is C:  
   
 def dispiay(self):  
 current =self.head  
   
 while current is not None;  
 print(current.data,end="");  
   
   
   
   
 }  
}

السؤال 3

public class m3 {  
   
 class Doubluelinlist:  
 def\_init(self):  
 self.head = data;  
 self.tail= None;  
 self.next = 0  
   
 class def{  
 def isEmpty(self):  
 reurnt self.size;  
 raise Exception("the list empty");  
 return self.head.data  
   
 def last(self):  
 if self.isEmpty():  
def size (self);  
 return self.size ;  
   
 def frist(self):  
 if self.isEmpty():  
   
 raise Eception(:the list is empty);  
   
 def last self:  
 if self.isEmpty():  
 raise Eception(:the list is empty);  
return self.tail.data;  
  
def addfirst (self, data);  
 new\_node = None(data):  
 if self isEmpty():  
   
 def addlast (self, data);  
 new\_node = None(data):  
 if self isEmpty():  
  
 def removefirst (self):  
 if self isEmpty():  
 self.head ==self.tail;  
 self.head =self.tail;  
 def removelast(self):  
 if self isEmpty():  
self.head ==self.tail;  
 self.head =self.tail;

السؤال 4

للعثور على العقدة الوسطي لقائمة مرتبطة بشكل مزدوج باستخدام طريقة التنقل الروابط.

1. قم بتهيئة مؤشرين إلى العقدة سيشر كلا المؤشرين في البداية إلى العقدة.
2. عند توقف الحلقة سيشر البطيء إلى العقدة الوسطى للقائمة المرتبطة بشكل مزدوج إذا كانت القائمة تحتوي على عدد زوجي من العقد تمثيل الكود الكاذب.
3. /\*/  
     
   findMiddleNode(list):  
   slowptr = list.header.next  
   real node  
    pastptr = list.header.next  
    Initialize fastptr to the first  
    real node  
   while fastptr.next!=  
   list.trailer and  
   fastptr.next.next  
   return slowptr  
     
    \*/

السؤال 5

public class m5 {  
  
 class def {  
 def\_init(self):  
  
 self.head =None;  
 self.tail=None;  
  
 def size(self)  
 count = 0  
 current = self.head  
   
 while current is not None:  
 count +=1  
 current=current.next  
 return count  
}  
}

السؤال 6

public class m6 {  
 private Node<T> head;  
 private Node<T> tail;  
 private int size;  
  
 public int getSize() {  
 return size;  
   
 if (this obj){  
 return true;  
 }  
 if (size);  
 return false;  
   
 while(courrentNode null);  
   
 private static class Node<T>;  
 private Node<T> previous;  
 private Node<T> next;  
 }  
}

السؤال 7

public class m7 {  
 class def{  
   
 }  
 def concatenate\_lists(L,M):  
 if L.next is None:  
 return M  
 if M.next is None:  
 return   
   
 L\_header = L.next   
 L\_trailer = L.prev  
 M\_header = M.next  
 M\_trailer = M.prev  
   
  
 L\_trailer.next =M\_header  
 M\_header.prev = L\_header   
 M\_trailer =.next  
 L\_header .prev = M\_trailer

السؤال 8

c class m8 {  
 class def{  
   
 }  
 def is\_empty(self):  
 return self.sentinel.next==self.sentinel  
   
 def insert\_at\_front(self,data):  
   
 def remove\_from-front(self)  
 if self.is\_isEmpty():  
 def remove\_from\_end(self):  
 if self.is\_isEmpty():

السؤال 9

class def{  
 def is\_Empty(self):  
 return self.head is None  
   
 def insert\_at\_end(self,data)  
 new\_node = Node(data)  
 if self.is\_Empty():  
 self.head=new-node;  
 self.head.next=self.head;  
   
 def intser\_at\_beginning (self,data);  
 self.head=new-node;  
 self.head.next=self.head;  
   
 def delete(self,data);  
 if self.is\_Empty():  
 return

السؤال 10

public class m10 {  
 class def{}  
   
 def append(self,data);  
 new-node = Node(data);  
   
 if self.head.is\_isEmpty():  
 self.head = new-node;  
   
 def clone(self):  
 if self.head.is\_isEmpty():  
 return  
 while currentl;  
 return clonde\_list:  
  
  
}