1. IDMTRE10: 3
2. IDEGRA02: Queue
3. IDMSQAS10: 1, 8, 10, -, -, -, 3
4. IDMAOA09: O(N^2)
5. IDMSQ08: No operation that has time complexity O(N)
6. IDESQ07: Add a new item to the queue at the rear position.
7. IDMSQAS11: Order of the elements of the list.
8. IDELI16: Abstract Data Type
9. IDETRE23: t.getLeftSubTree()
10. IDESQAS13: 4199 and 9679 hash to the same value
11. IDMSQ03: pop()-->pop()-->pop()-->push("2")-->push("3")-->push("1")
12. IDHLI05: beforeTail.setNext(null); tail.setNext(head); head=tail;
13. IDESQAS14: Binary search is faster than linear search, but it requires a sorted array.
14. IDEGRA03: The weight of the shortest path from vertex Vi to vertex Vj using intermediate verties in the set {V1..Vk}.
15. IDMAOA03: O(N)
16. IDESOA09: In a min-heap the parent node value is always greater than or equal to its children's values.
17. IDHTRE07: E
18. IDHTRE01: Post-order.
19. IDETRE22: t.getRightSubTree()
20. IDETRE07: This is a binary search tree.
21. IDEGRA07: Unweighted, undirected, complete graph
22. IDETRE15: p[node]
23. IDELI14: X(data, prev, next)
24. IDMAOA05: O(N)
25. IDEGRA08: Weigh of an edge must be possitive.
26. IDESOA10: The input array is divided into two parts at the middle of the array.
27. IDESOA01: The sort key must be numeric.
28. IDESQ16: X
29. IDESQAS01 Linear time
30.
31. IDMGRA01: N-1
32. IDETRE13: Node G.
33. IDESOA16: Merge sort.
34. Method search() is used to search for an item in a singly linked list. Please complete the code for this method?

```
    public int search(int data)
    {
        int count=1;
        SLNode current=this.head;
        while ((current !=null) && (current.getData()
!= [data] ))
        {
            count++;
            current= [current.getNext()] ;
        }
```

```
        if (current == null)
            return -1;

    else

        return count ;

}
```

35. IDETRE14: The parent node of node K.
36. IDMTRE08: D,B,J,G,E,H,I,F,C,A
37. IDESQAS05: Two entries with different keys have the same exact hash value.
38. IDMSQ01: S={“A”,“B”,“C”,“D”}
39. IDEGRA01: 2E.
40. IDESQAS04: The array must be sorted.
41. IDEGRA04: Parallel edges.
42. IDMTRE01: Descending order.
43. IDMTRE17: Value of node C is smaller than value of node A and node B.
44. IDHAOA08: O(N^4)
45. IDMTRE21: 21
46. IDHTRE02: One node.
47. IDETRE19 : preOrderTraversal(getLeftChild(node))
48. IDETRE08: This is an expression tree.
49. IDHSOA05: T(N)=T(N/3)+T(2N/3)+O(N)
50. IDETRE03: Complete binary tree.
51. IDESOA14: O(P(N+B))
52. IDESOA05: Based on Divide and Conquer approach.
53. IDESOA12: A stable sorting algorithm is used to sort the digits.
54. IDMTRE20: Node C has the biggest value
55. IDMSQAS01: 8 2 + 5 7 + * 10 - 9 * 3 +
56. IDMSOA08: A={2,5,9,8,10,13,12,22,50}
57. IDMLI09: ‘E’-->‘C’-->‘A’
58. IDEGRA09: Queue
59. IDELI13: head
60. IDMGRA02: Parallel edges
61. IDMSQ11: m=m-1
62. IDETRE21: postOrderTraversal(getRightChild(node))
63. IDMSQAS02: + 5 * + 7 * 9 3+ 2 8
64. IDETRE04: 2^h.
65. IDESOA15: Sorting
66. IDETRE10: If an interior node has two children, then this node’s label must be an operator.

67. IDESQ11: Queue is empty when front=rear.
68. IDESQAS07: 512.
69. IDETRE12: Node C.
70. The following method reverses the item's order of a stack using a queue. Please complete the code of the method?

```
public static int reverse(SLLStack s)
{
        ArrayQueue q = new ArrayQueue();
        while (!s.isEmpty())
        {
            StackNode node = [s.pop()] ;
            q.enqueue(node.getData());
        }
        while ( [!q.isEmpty()] )
        {
            StackNode newnode = new StackNode(q.dequeue());
            s.push([new noc] );
        }
}
```

71. IDEGRA10: Adding a vertex in adjacency matrix representation is easier than adjacency list representation.
72. IDHAOA05: O(N)
73. IDMSOA04: Merge sort
74. IDMSQAS09: A
75. IDMSQ12: 5
76. IDHTRE10: DECBUTZYXA
77. IDESQ06: Remove an item from the queue at the front position
78. IDMSQAS04: 150
79. IDMTRE12: Min heap.
80. IDMGRA04: P, Q, R, U, S, T
81. IDMSQAS05: Print binary representation of n.
82. IDESQAS10: 6.
83. IDESQAS15: middle=(left + right)/2
84. IDESQ14: 40
85. IDMSQAS08: B
86. IDESQ15 rear
87. IDMTRE18: Value of node C is bigger than value of node B, but smaller than value of node A.
88. IDEGRA06: A matrix contains only 0 and 1.
89. The following method reverses the item's order of a stack using a queue. Please complete the code of the method?

```
public static int reverse(SLLStack s)
{
        ArrayQueue q = new ArrayQueue();
        while (!s.isEmpty())
        {
            StackNode node = [s.pop()] ;
            q.enqueue(node.getData());
        }
        while ([!q.isEmpty()] )
        {
            StackNode newnode = new StackNode(q.dequeue());
            s.push([new noc] );
        }
}
```

90. IDMLI11: ‘F’-->‘D’-->‘B’
91. IDESOA04: The relative order of elements with equal keys are maintained.
92. IDHAOA03: O(N^3)
93. IDESQAS11: i.
94. IDMTRE13: 4
95. IDELI02: Integer.
96. IDMSQ07: Queues use two ends of the structure; stacks use only one.
97. IDESQ13: 10
98. IDMGRA03: Performing a BFS starting from S
99. This method implement an O(N) algorithm to rearrange array x so that the left part is the elements that is smaller than p, the right part is the elements that is bigger than p. Please complete the code for this method?

```
public static void rearrange(int [] x, int p)
{
int left=0;
int right=x.length-1;
while ([left<right] )
{
while ((x[left]<p)&&(left<x.length))
[left++] ;
while ((x[right]>p)&&(right>=0))
[right--] ;
if (left<right)
{
int tmp=x[left];
x[left]=x[right];
x[right]=tmp;
```

```
}
}
}
```

IDEAOA05:Theoretical approach

IDEAOA01: Algorithm is a step-by-step procedure for solving a problem in a finite amount of time

IDESQ02: Dequeue is a special type of queue

IDMLI03: remove one item from the list

IDELI07: null

IDESOA03: The sort key must be numeric

IDMLI04: Remove the node at the pos position from the list

IDMSOA12: C={3,9,10,27,38,43,82}

IDESOA06: Selection sort

IDMAOA04: O(N)

IDESQAS12: loại đáp án =]]z hi(k)=h(k) mod N.

IDELI04: Array-based is faster than linked-list in case of accessing list's items.

IDELI03: Boolean.

IDMTRE06: A,B,D,C,E,G,J,F,H,I

IDMSQ02: Q={"D", "E" ,"F" ,"D"}

IDMSOA03: A={8,23,32,45,56,78}

IDETRE17: The left child and right child of node i are 2i+1 and 2i+2

IDHTRE06: (1 (2 3 4) (5 6 7))

IDMTRE07: B,D,A,G,J,E,C,H,F,I

IDETRE16: l[node]

IDESQ03: add an item to the stack.

IDESOA08: Bubble sort

IDESQAS08: Evaluating a posfix expression

IDELI11: Y.getNext().

The method below represent a number k in base b using a stack. Please complete the code of this method? (đúng ½ code -_-)

```
public void BaseConversion(int k, int b)
{
        ArrayStack s = new ArrayStack();
        while (k/b != 0)
        {
            s.push(k%b);
            k=k/b;
        }
        s.push(k);
        while (!s.isEmpty())
```

```
            System.out.print( s.pop() );
    }
```

125. IDEGRA05: A symmetric matrix over its diagonal.

IDESQ09 rear=(rear+1)%maxSize

126. Method search() is used to search for an item in a singly linked list. Please complete the code for this method?

```
    public int search(int data)
    {
       int count=1;
        SLNode current=this.head;
        while ((current !=null) && (current.getData()
!= data ))
        {
            count++;
            current= current.getNext() ;
        }
        if (current == null)
            return -1;
        else
            return count ;
    }
```

127. IDMSQAS03: 129

128. IDEAOA15: They are mathematic notation for comparing growth rates between functions

129. Method search() is used to search for an item in a singly linked list. Please complete the code for this method?

```
public int search(int data)
{
        int l=getLength();
        for (int i=1; i<l; i++)
        {
            SLNode aNode= aNode. ;
            if (aNode.getData()==data)
                return i;
        }
        return 0 ;
}
```

130. The following method implement the recursive version of the binary search algorithm. Please complete the code of the method?

```
    public static int BinarySearch(int []a, int key, int
left, int right)
    {
        if (left > right)
            return KE ;
        else
        {
            int mid = (left + right)/2;
            if ( a[mid]<key )
               return BinarySearch(a,key,mid+1,right);
            else
            {
               if (a[mid]>key)
                 return BinarySearch(a,key,left, mid-1 );
               else
                 return mid;
            }
        }
    }
```

131. Please complete the code of the linear search method below?

```
  public int LinearSearch(int[] a, int key)
  {
        int index=0;
        boolean found=false;
        int pos=-1;
        while ((index<n)&&(!found)
        {
            if ( a[index]!=key )
                        {
                             found=true;
                             pos=index;
                        }
                        index++;
        }
                 return pos ;
  }
```

132. Method swap() is used to swap two nodes in a Singly Linked List. Please complete the code for this method?

```
  public void swap(int pos1, int pos2)

  {

      SLNode node1 = get(pos1);
```

```
        SLNode node2 = get(pos2) ;
        SLNode tmp=new SLNode(node1.getData());
        node1.setData( node2.getData() );
        node2.setData( tmp.getData() );
    }
```

133.