IDEAOA02

- Which is not a property of a algorithm?

Select one:

- It must terminate for all inputs.
- The order of the algorithm's steps must be precisely defined.
- The execution time and the memory needed for a algorithm must be percisely defined.
- It must be correct and composed of precisely defined steps.

IDEAOA03

– Which statement below is wrong?

Select one:

- For the same data, some data structures may require more or less space.
- A data structure is a way of organizing data for processing within a computer program.
- A data structure is a piece of information (a physical instantiation of a data type)
- For the same operations on the data, some data structures lead to more or less efficient algorithms.

IDEAOA04

– Which statement is correct concerning to the complexity of algorithm?

Select one:

- The complexity of an algorithm is determined by the total lines of code of the program that implements the algorithm using a given programming language.
- The complexity of an algorithm is a measure of the amount of time and cost needed to implement this algorithm.
- The complexity of an algorithm is a measure of the amount of time and space required by the algorithm for an input of a given size n.
- The complexity of an algorithm is determined by the maximum value of the input size n that does not affect the correctness of the algorithm.

IDEAOA05

– When evaluating algorithm's complexity, which approach makes possible an evaluation that is independent of the hardware and software environments?

Select one:

- Using input data sets of varying size.
- Theoretical approach.
- Measuring the running time and memory space using the same hardware and software environment.
- Experimental approach.

IDEAOA06

– What is time complexity of an algorithm?

Select one:

- The amount of time needed to implement the algorithm.
- The amount of time that the algorithm needs to run for an input of a given size n.
- The upper limits for excution time of the algorithm.
- The response time of the algorithm.

IDEAOA08

– Which statement is wrong concerning to the best-case time complexity of an algorithm?

Select one:

- The best case of an algorithm A is estimated as the minimum number of primitive operations performed by A on an input size n.
- Many algorithms perform exactly the same in the best case.
- The best-case is used frequently to analyze the time complexity of algorithms.
- The best-case gives us an lower bound on the time complexity of algorithms.

IDEAOA09

– Which statement is wrong concerning to the average-case time complexity of an algorithm?

Select one:

- The average-case is places somewhere between the best-case and the worse-case.
- The average-case of an algorithm A is estimated as the average number of primitive operations performed by A on an input size n.

- [x] The average-case of an algorithm A is depended on the characteristic of the input data.
- [ ] The average-case is easy to determine.

IDEAOA10

– Which statement is wrong concerning to the worst-case time complexity of an algorithm?

Select one:

- [ ] The worst case gives us an upper bound on time complexity of an algorithm.
- [x] The worst-case of an algorithm A is estimated as the maximize number of primitive operations performed by A on an input size n.
- [ ] At the worst-case the algorithm takes more time to finish than it does at the average-case and best-case.
- [ ] The worst-case is not very informative because many algorithms rarely perform at their worst-case.

IDEAOA11

– Which one determines the asymptotic behavior of the function T(n)?

Select one:

- [x] The leading term.
- [ ] The term has the biggest coefficient.
- [ ] The first term.
- [ ] The last term.

IDEAOA12

– Which notation represents the upper-bound of the grow rate of a function?

Select one:

- [ ] Big-Theta notation
- [ ] Big-Omega notation
- [x] Big-Oh notation
- [ ] Big-Alpha notation

IDEAOA14

– Suppose that the estimated time complexity of algorithm A and algorithm B is TA(N) and TB(N) respectively. How can we compare the time complexity of A and B?

Select one:

- [ ] We compare the value of TA and TB corresponding to some special value of n.

- We compare the value of TA and TB corresponding to a very large, pre-defined value of n.
- We compare the value of TA and TB corresponding to every value of n.
- We compare the grow rate of the leading terms of TA(N) and TB(N).

IDELI01

- Which statement below is wrong in the context of list data structure?

Select one:

- In the list, items are referenced by their value.
- Every item in the list, except for the head and tail, has an unique predecessor and an unique successor.
- List can be implemented using an array or a collection of linked nodes.
- A list is a sequence of zero or more items of the same type.

IDELI02

– In the ADT of the list data structure, getLength() method returns a/an _____ value?

Select one:

- Real number.
- Boolean.
- String.
- Integer.

IDELI03

– In the ADT of the list data structure, isEmpty() method returns a/an _____ value?

Select one:

- String.
- Real number.
- Boolean.
- Integer.

IDELI04

– Which statement is correct about array-based list?

Select one:

- Array-based is faster than linked-list in case of accessing list's items.
- They can be implemented by Java language only.
- Array-based is faster than linked-list in case of inserting new item into the list.
- Elements of array-based list can be located dinamically and discontinuously.

IDELI05

– Which statement is correct about linked-list?

Select one:

- Linked-list cost more than array-based list in term of deleting and inserting operations.
- Elements of linked-list can be located dinamically and discontinuously.
- Elements of linked-list must be stored in consecutive memory blocks.
- Array-based list is more flexible in list's size than linked-list.

IDELI06

– In the ADT of the list data structure, remove(int post) method will?

Select one:

- Remove an item at the pos position form the list.
- Remove the last item form the list.
- Remove the first item from the list.
- Remove all items from the list.

IDELI07

– In a single linked-list, if a NodeX(data,next) is a tail which is the value of the X's next?

Select one:

- undefined
- null

- head
- 0 <not sure>

IDELI08

– In a Singly Linked List that have only one node X, which value does X.getNext() return?

Select one:

- null
- The tail node
- The head it self
- undefined

IDELI09

– Suppose that X is a node in the middle of the Singly Linked List. Complete the code below to delete all nodes after X from the list?X.setNext(_____);

Select one:

- null.
- X.getNext().
- X.getNext().getNext().
- tail.

IDELI10

– Suppose that X is a node in the middle of the Singly Linked List. Complete the code below to delete one node after X from the list?X.setNext(_____);

Select one:

- null.
- X.getNext().
- X.getNext().getNext().
- tail.

IDELI11

– Complete the code below to insert a new node X at the POS position of a Singly

Linked List?

```
SLNode Y=traversing(POS); //travel to POS position
X.setNext( _______ );
Y.setNext(X);
```

Select one:

- Y.setNext(tail).
- Y.getNext(). (selected)
- X.getNext().
- X.setNext(Y).

IDELI12

– Complete the code below to travel from the head node to the POS position of a Singly Linked List?

```
int c=1;
SLNode node = head;
while (c < POS)
{
    ________;
    node=node.getNext();
}
return node;
```

Select one:

- node++
- node=node.setNext(tail)
- c=c+1
- c=c.getNext() (selected)

IDELI13

– In a Circle linked-list, if a NodeX(data,next) is a tail which is the value of the X's next?

Select one:

- undefined

- null
- head
- 0

IDELI14

– Which is the common form of a node X in a Doubly Linked List?

Select one:

- X(data, next)
- X(data)
- X(data, prev, next)
- X(data, prev)

IDELI15

– Which is common form of a node X in a Singly Linked List?

Select one:

- X(data, next)
- X(data)
- X(data, prev, next)
- X(data, prev)

IDELI16

– A mathematical-model with a collection of operations defined on that model is called?

Select one:

- Primitive data type.
- Data structure.
- Algorithm.
- Abstract Data Type.

IDESOA02

- Which statement below is wrong about comparison sorting algorithms?

Select one:

- The time complexity of some comparison sorting algorithms can be faster than O(NlogN).

- Bubble sort, Merge sort and Heap sort are comparison sorting algorithms.
- The time complexity of a comparison sorting algorithm is based on the number of comparisons and moves during sorting.
- The sorted order is determined based only on the comparisons between sort keys.

IDESOA03

– Which statement below is wrong in the context of linear sorting algorithm?

Select one:

- The time complexity is linear.
- The sorted order is determined based on the comparisons between sort keys.
- The sort key must be numeric.
- Counting sort and Radix sort are linear sorting algorithms.

IDESOA04

– In a stable sort algorithm …?

Select one:

- The relative order of elements with equal keys are not maintained.
- The order of both key and non-key values are maintained.
- The relative order of elements with equal keys are maintained.
- The order of key values are maintained.

IDESOA05

– Merge sort and Quick sort are …?

Select one:

- Based on Divide and Conquer approach.
- O(n^2) sorting algorithms.
- Linear sorting algorithm.
- The fastest sorting algorithms.

IDESOA06

– Which sorting algorithm locates the largest (or smallest) key and its index in each sort pass?

Select one:

- Bubble sort.

- Selection sort.
- Insertion sort.
- Heap sort.

IDESOA07

– Which statement is wrong about Insertion sort?

Select one:

- Scan and exchange any pair of elements that is out-of-order.
- It is O(n^2) sorting algorithm.
- Unsorted elements are inserted into an already sorted list.
- We must shift several elements to make place for the inserted one.

IDESOA08

– Which sorting algorithm scans and exchanges any pair of elements that is out-of-order?

Select one:

- Insertion sort.
- Bubble sort.
- Selection sort.
- Heap sort.

IDESOA09

– Which statement is wrong concerning to the Heap data structure?

Select one:

- It is a tree where all nodes have zero, one or two children.
- In a min-heap the parent node value is always greater than or equal to its children's values.
- It is used in Heap sort algorithm.
- An array can be used to store heap's nodes.

IDESOA10

– In Merge sort algorithm…?

Select one:

- .The input array is divided into two parts at the middle of the array
- The input array is divided into two parts based on the pivot values
- The merge algorithm combines two sorted array by attaching the second array to the end of the first one
- The worst case is O(N^2)

IDESOA011

– Which statement is wrong about Quick sort?

Select one:

- A merge algorithm is needed to combine two partitioned arrays
- The worst case is O(N^2)
- The array is partitioned based on the pivot value
- By choosing the pivot carefully, we can reduce the running time of the algorithm.

IDESOA14

– Suppose that we are using Radix sort on N elements, each element has P digits in base b (each digit is in the range [0 .. B-1]), and couting sort algorithm is used to sort the digits. What is the time complexity of the Radix sort algorithm?

Select one:

- O(N.P.B).
- O(P+N+B).
- O(B+N).
- O(P(N+B)).

IDESOA15

– What is an operation in which a list of elements is arranged either in ascending order or in descending order?

Select one:

- Searching.
- Hashing.
- Sorting.
- Traversing.

IDESOA16

– Which of the following sorting algorithm does not have a worst case time complexity of O(n^2)?

Select one:

- Insertion sort.
- Quick sort.
- Buble sort.
- Merge sort.

IDESQ01

- Which statement below is wrong concerning to stack data structure?

Select one:

- It is a First In First Out (FIFO) list.
- A stack contains a sequence of zero or more items of the same type.
- push() and pop() are two operations defined in Stack's ADT.
- List-based stack has no limit on total number of items of the stack.

IDESQ02

– Which statement below is wrong about queue data structure?

Select one:

- Dequeue is a special type of queue.
- It is a First In First Out (FIFO) list.
- Queue can be implemented using an array or a linked-list.
- enqueue() and dequeue() operations must be performmed at one end of the queue.

IDESQ03

– In the ADT of the Stack data structure, push() method is used to

Select one:

- take an item out of the stack
- get the total items in the stack
- get an item without deleting it from the stack
- add an item to the stack

IDESQ04

– Which statement is correct about array-based stack?

Select one:

- To add a new item into the stack: firstly, top is increased by 1, then current items will be shifted one slot to the right to make space for the new item.
- To add a new item into the stack: firstly, top is increased by 1, then current items will be shifted one slot to the left to make space for the new item.
- top is the first item of the array.
- top is the last item of the array.

IDESQ06

– In the ADT of the Queue data structure, dequeue() method will?

Select one:

- Remove an item from the queue at the rear position.
- Add a new item to the queue at the front position.
- Add a new item to the queue at the rear position.
- Remove an item from the queue at the front position.

IDESQ07

– In ADT of the Queue data structure, enqueue() method will?

Select one:

- Add a new item to the queue at the front position.
- Remove an item from the queue at the front position.
- Remove an item from the queue at the rear position.
- Add a new item to the queue at the rear position.

IDESQ08

– Which statement is wrong about array-based circular queue?

Select one:

- when front=rear the queue is empty.
- rear can be wrap around to the beginning of the array.
- front can be wrap around to the beginning of the array.
- when front=rear the queue is full.

IDESQ09

– Complete the code for the enqueue() method in array-based circular queue?

Select one:

- rear=rear+1
- rear=(rear+1)%maxSize
- front=front+1
- front=(front+1)%maxSize

IDESQ10

– Complete the code for the dequeue() method in array-based circular queue?

```
public void dequeue()
{
  if (!isEmpty())
  {
       int pos=front;
       ________;
       return items[pos]
   }
}
```

Select one:

- front=front+1
- rear=rear+1
- front=(front+1)%maxSize
- rear=(rear+1)%maxSize

IDESQ11

– Which statement is wrong about list-based queue?

Select one:

- front is the head and rear is the tail of the linked-list.

- A linked-list is used to implement the queue.
- Queue is empty when front=rear.
- List-based queue ADT does not have isFull() operation

IDESQ12

– Which of the following statement is true?

Select one:

- In both array-based stack and queue, when removing an item the corresponding index is increase by 1
- The top of a stack corresponds to the front of a queue
- The contents of a queue can wrap around , while those of a stack can not
- .The pop() operation on a stack is simpler than the dequeue() operation on a queue.

IDESQ13

– Suppose you push 10, 20, 30, 40 onto a stack, then you pop three items. Which one is left on the stack?

Select one:

- 40
- 30
- 10
- 20

IDESQ14

– Suppose you enqueue 10, 20, 30, 40 onto a queue, then you dequeue three items. Which one is left on the queue?

Select one:

- 40
- 30
- 10
- 20

IDESQ15

– The end which a new element gets added to a queue is called

Select one:

- Bottom
- Top
- Rear
- Front

IDESQ16

– What is the result of the following operation on the stack S: S.peek(S.push(X))?

Select one:

- X.
- S.push(X).
- Null.
- S.top.

IDESQAS01

– What is the worst-case time for linear search finding a single item in an array?

Select one:

- Logarithmic time.
- Linear time.
- Constant time.
- Quadratic time.

IDESQAS03

– What is the worst-case time for finding a key in a hash table (assume that there is no collision)?

Select one:

- Constant time.

- (●) Linear time.
- ( ) Quadratic time.
- ( ) Logarithmic time.

IDESQAS04

– What additional requirement is placed on an array, so that binary search may be used to search for a key?

Select one:

- ( ) The array elements must form a heap.
- (●) The array must be sorted.
- ( ) The array must have at least 2 entries.
- ( ) The array's size must be a power of two.

IDESQAS05

– What is the best definition of a collision in a hash table?

Select one:

- (●) Two entries with different keys have the same exact hash value.
- ( ) Two entries are identical except for their keys.
- ( ) Two entries with the exact same key have different hash values.
- ( ) Two entries with different data have the exact same key.

IDESQAS06

– A separate chaining hash table has an array size of 512. What is the maximum number of entries that can be placed in the table?

Select one:

- ( ) There is no maximum.
- ( ) 256.
- ( ) 1024.
- ( ) 512.
- (●) 511.

IDESQAS08

– Which of the following is not an application of the queue data structure?

Select one:

- Job scheduling
- Evaluating a posfix expression
- Stack reversing
- Packet queueing

IDESQAS10

– Consider a hash table of size seven, with starting index zero, and a hash function h(k)=(3k+4) mod 7. What is the address of the key k=10?

Select one:

- 3.
- 0.
- 7.
- 6.

IDESQAS11

– Complete the code below to search for key in an array using linear seach algorithm?

Select one:

- -1.
- true.
- a[i].
- i.

IDESQAS12

– In a hash table of the size N using linear probing, what is the probing hash function hi(k)?

Select one:

- hi(k)=(h(k)+i) mod N.
- hi(k)=i mod N.
- hi(k)=h(k) mod N.
- hi(k)=i + k.

IDESQAS13

– Given the following input (4322, 1334, 1471, 9679, 1989, 6171, 6173, 4199) and the hash function: h(k)=k mod 10. Which of the following statements are true?

Select one:

- All elements hash to the same value.
- 1471 and 6171 has to different value.
- Each element hashes to a different value.
- 4199 and 9679 hash to the same value.

IDESQAS14

– In the context of seach algorithms, which of the following statements are true?

Select one:

- Binary search is the fastest search algorithm.
- Linear search is faster than binary search.
- Hash data structure is used to support sorting.
- Binary search is faster than linear search, but it requires a sorted array.

IDESQAS15

– Which of the following statements is used in binary search algorithm to halve the array?

Select one:

- middle=middle/2
- middle=(right-left)/2
- middle=(left + right)/2
- middle=midlle*2

IDHLI01

– Suppose that you want to sort a singly linked list, each list's item is a large object. which of the following sort algorithms should be used to minimum the time complexity?

Select one:

- Quick sort.
- Insertion sort.

Heap sort.

Bubble sort.

IDHLI02

– Method reverse() below is used to reverse the order of items in a Singly Linked List. Please complete the code of the method?

```
int maxSum=0;
for (int i=0; i<a.length; i++)
  for (int j=i; j<a.length; j++)
  {
    int thisSum=0;
    for (int k=i; k<=j; k++)
      thisSum+=a[k];
    if (thisSum>maxSum)
      maxSum=thisSum;
  }
return maxSum;
```

Select one:

prev=head

head=current

current=head

head=prev

IDHLI03

– Consider a Singly Linked List contains N nodes (N > 8), a method f1() is designed to find the 8th node from beginning, and method f2() is designed to find the 8th node from end. Which is the time complexity of f1() and f2()?

Select one:

O(N) and O(1)

O(N) and O(N)

O(1) and O(N)

O(1) and O(1)

IDHLI04

– Method deleteTail() below is used to delete the last node in a Singly Linked List. Please complete the code of the method?

```
public void deleteTail()
{
  int pos = getLength();
  SLNode beforeTail=traversing(pos-1);
  __________;
}
```

Select one:

- beforeTail.setNext(null)
- beforeTail = tail
- beforeTail.setNext(tail)
- beforeTail = null

IDHLI05

– Method tailToFront() below moves the last node of a Singly Linked List into the front of the list. Please complete the code of the method?

```
public void tailToFront()
{
  int pos = getLength();
  SLNode beforeTail=traversing(pos-1);
  SLNode tail=beforeTail.getNext();
  ________
}
```

Select one:

- beforeTail.setNext(null); tail.setNext(head); head=tail;
- tail.setNext(null); beforeTail.setNext(head); head=beforeTail;
- tail.setNext(head); head.setNext(null); beforeTail=head;
- head.setNext(null); beforeTail.setNext(tail); tail=beforeTail;

IDHSOA01

– Selection sort is used to sort an array in the descending order. When does the worst case occur?

Select one:

- The array has several duplicated items.
- The array is already sorted in the ascending order.

- The array is already sorted in the descending order.
- The first and the last items of the array are the same.

IDHSOA02

– Insertion sort is used to sort an array in the descending order. When does the best case occur?

Select one:

- The array contains several zero items.
- The array is already sorted in the descending order.
- The array is already sorted in the ascending order.
- The array has several duplicated items.

IDHSOA03

–Given an array A that is almost sorted (only one or two elements are misplaced). Which sorting algorithm gives the best time efficiency when applied on A?

Select one:

- Quick sort
- Bubble sort
- Merge sort
- Insertion sort

IDHSOA04

– Which of the following sorting algorithms has the minimum number of swap operations in general?

Select one:

- Heap sort
- Insertion sort
- Quick sort
- Bubble sort

IDHSOA05

– Consider a modified version of Merge sort where the input array is partitioned at the position one-third of the length N of the array. What is the recurrence of this algorithm?

Select one:

T(N)=2T(2N/3)+O(N)

T(N)=2T(N/3)+O(N)

T(N)=T(N/3)+T(2N/3)+O(N)

T(N)=T(N/2)+T(3N/2)+O(N)

IDHSOA06

– Which of the following sorting algorithms has the lowest worst case time complexity?

Select one:

Quick sort

Bubble sort

Merge sort

Insertion sort

IDHSOA07

– Consider an array A where the items are in the range from 1 to n^3. Which of the following sorting algorithms gives the best time efficiency when applied on A?

Select one:

Radix sort

Counting sort

Quick sort

Heap sort

IDHSQAS01

– What data structure would you mostly likely see in a non-recursive implementation of a recursive algorithm?

Select one:

Tree.

Queue.

Stack.

Linked-List.

IDHSQAS03

– Which of the following is a hash function?

Select one:

- Folding
- Quadratic probing
- Chaining
- Hashing
- Open addressing

IDHSQAS04

– If h(k) is any hash function and is used to hash N keys into a table of size M, where N < M, the expected number of collisions involving a particular key X is?

Select one:

- Less than M
- Less than N
- Less than (N+M)/2
- Less than 1.

IDHSQAS05

- The process of accessing data stored in a serial access memory is similar to manipulating data on a?

Select one:

- Queue.
- Binary Tree.
- Stack.
- Heap.

IDMAOA01

– What is the time complexity of the following algorithm with respect to the input size N

```
Algorithm: PolyEvaluate(C,n,x)
Input: An array C stores n real numbers that are the
coefficients of the polynomial. A real value x.
Output: The value of f(x).

S←0
For i ← 0 to n-1 do
{
  P←1
  If (i <> 0)
    For (k ← 1 to i) do
       P←p*x
  S←s+C[i]*p
}
Return s;
```

Select one:

- O(N)
- O(2N)
- O(N^2)
- O(1)

IDMAOA02

– What is the time complexity of the following algorithm with respect to the input size N

```
Algorithm:Search(A,n,m)
Input: An array A stores n sorted integer. An integer m.
Output: An integer i that a[i]=m.
        -1 if m doesn't appear in the array.

For i ← 0 to n-1 do
  If (a[i] = m) then
    Return i;
```

Select one:

- O(N^2)
- O(N-1)
- O(N)
- O(2N)

IDMAOA03

– What is the time complexity of the following algorithm with respect to the input size N

```
Algorithm: GCD(m,n)
Input: Two integers m and n
Output: The value of gcd(m,n)

i←n
While (i>1) do
  If (m is divisible by i) AND (n is divisible by i)
    Return i
  i--
Return 1
```

Select one:

- O(M)
- O(N+M)
- O(N)
- O(N*M)

IDMAOA05

– What is the time complexity of the following algorithm with respect to the input size N

```
Algorithm sum(n)
Input: an integer n
Output: the sum S = ∑_{i=1}^{n} i^3

     s ← 0
     for  i← 1 to n do
          s=s+i*i*i;
     return s;
```

Select one:

- O(1)
- O(N^2)
- O(2N)
- O(N)

IDMAOA06

– Estimate the time complexity in Big-Oh notation, with respect to the input size N, for the code below:

```
public void reverse()
{
  SLNode prev=null;
  SLNode current = head;
  SLNode tmp;
  while (current !=null)
  {
    tmp=current.getNext();
    current.setNext(prev);
    prev=current;
    current=tmp;
  }
  __________;
}
```

Select one:

- O(N^2)
- O(1)
- O(N)
- O(N^3)

IDMAOA07

– Estimate the time complexity in Big-Oh notation, with respect to the input size N for the code below

Select one:

- O(N^1000)
- O(N)
- O(1)
- O(1000N)

IDMAOA08

– Estimate the time complexity in Big-Oh notation, with respect to the input size N for the code below

```
for (i = 0; i < N; i++)
     for (j = 0; j < N * N; j++)
      sum++;
```

Select one:

- O(1)
- O(N^2)
- O(N^3)
- O(N)

IDMAOA09

– Estimate the time complexity in Big-Oh notation, with respect to the input size N for the code below

Select one:

- O(N)
- O(N^3)
- O(N^2)
- O(1)

IDMAOA10

– Estimate the time complexity in Big-Oh notation, with respect to the input size N for the code below

```
for (i = 1; i < N; i++)
   for(j = 1; j < i * i; j++)
        sum++
```

Select one:

- O(N^2)
- O(N)
- O(1)

O(N^3)

IDMAOA11

– Estimate the time complexity in Big-Oh notation, with respect to the input size N for the code below

```
for (i = 0; i < n; i++)
     for (j = 0; j < i * i; j++)
       for (k = 0; k < j; k++)
          sum++;
```

Select one:

O(N^4)

O(N^2)

O(N^5)

O(N^3)

IDMAOA12

– Consider two algorithms which have the time complexity in Big-Oh notation below. Which statement is correct?

Select one:

Two algorithm are equivalent in term of time efficiency.

The second algorithm is four times faster than the first one.

With the same value of N, two algorithm have the same computational steps.

The first algorithm is four times faster than the second one.

IDMAOA13

– Consider three algorithms which have the time complexity in Big-Oh notation below. Please arrange these algorithms in the ascending order of time efficiency (the slowest algorithm is the first one in the order).

$f_1(N)=O(2N^5 + N)$; $f_2(N)=O(N\log N)$; $f_3(N)=O(2^N)$

Select one:

- Algorithm 1, Algorithm 3, Algorithm 2
- Algorithm 1, Algorithm 2, Algorithm 3
- Algorithm 3, Algorithm 2, Algorithm 1
- Algorithm 2, Algorithm 3, Algorithm 1
- Algorithm 2, Algorithm 1, Algorithm 3
- Algorithm 3, Algorithm 1, Algorithm 2

IDMAOA14

– Consider three algorithm which have the time complexity in Big-Oh notation below. Please arrange these algorithms in the descending order of time efficiency (the fastest algorithm is the first one in the order).

$f_1(N)=O(N\log N);\ f_2(N)=O(\log N);\ f_3(N)=O(N)$

Select one:

- Algorithm 2, Algorithm 3, Algorithm 1
- Algorithm 1, Algorithm 2, Algorithm 3
- Algorithm 2, Algorithm 1, Algorithm 3
- Algorithm 1, Algorithm 3, Algorithm 2
- Algorithm 3, Algorithm 2, Algorithm 1
- Algorithm 3, Algorithm 1, Algorithm 2

IDMAOA15

– Estimate the time complexity in Big-Oh notation, with respect to the input size N for the code below

```
int x=0;
for (i=0; i<n; i++)
{
      int j=n/2;
      while (j>0)
      {
            x=x*i;
            j--;
      }
}
return x;
```

Select one:

- O(2N)
- O(N)
- O(N^2)
- O(N/2)

IDMLI02

– In a Singly Linked List implementation, what do we do when assigning head to null?

Select one:

- Avoid traversing the list.
- Delete all nodes from the list.
- Remove the last node from the list.
- Remove the first node from the list.

IDMLI03

– In an Array-based list, what does this code do to the list?

```
for (int i=pos-1;i<length;i++)
  items[i]=items[i+1];
length--;
```

Select one:

- Remove all item from the list except one.
- Traversing the list.
- Remove an item from the list.
- Duplicate items in the list.

IDMLI04

– In a Singly Linked List implementation, what does this code to to the list?

```
if (!isEmpty())
{
 if (pos == 1)
  head=head.getNext();
 else
 {
  SLNode prevNode=traversing(pos-1);
  SLNode posNode=prevNode.getNext();
  prevNode.setNext(posNode.getNext());
 }
}
```

Select one:

- Search for a node in the list
- Remove the tail node
- Remove the node at the pos position from the list
- Remove the head node

IDMLI05

– In an Array-based list, what does this code do to the list?

```
if (length<maxSize)
{
  length++;
  for (int i=length-1; i>pos-1; i--)
    items[i]=items[i-1];
  items[pos-1]=newItem;
}
```

Select one:

- Insert an item to the list
- Search for an item in the list
- Traversing the list
- Remove an item from the list

IDMLI06

– Consider method F in Java and a singly linked list L below. Suppose that H is the head node of the list L. What is the result if we call F(H)?

Select one:

- ‘B’-->‘D’-->‘F’
- ‘A’-->‘B’-->‘C’-->‘D’-->‘E’-->‘F’
- ‘A’-->‘C’-->‘E’
- ‘F’-->‘E’-->‘D’-->‘C’-->‘B’-->‘A’

IDMLI07

– Consider method F in Java and a singly linked list L below. Suppose that H is the head node of the list L. What is the result if we call F(H)?

Select one:

- ‘A’-->‘B’-->‘C’-->‘D’-->‘E’-->‘F’
- ‘A’-->‘C’-->‘E’
- ‘B’-->‘D’-->‘F’
- ‘F’-->‘E’-->‘D’-->‘C’-->‘B’-->‘A’

IDMLI08

- Consider method F in Java and a singly linked list L below. Suppose that H is the head node of the list L. What is the result if we call F(H)?

Select one:

- ‘E’-->‘C’-->‘A’
- ‘F’-->‘D’-->‘B’
- ‘A’-->‘C’-->‘E’
- ‘B’-->‘D’-->‘F’

IDMLI09

- Consider method F in Java and a singly linked list L below. Suppose that H is the head node of the list

L. What is the result if we call F(H)?

Select one:

- ‘B’-->‘D’-->‘F’
- ‘E’-->‘C’-->‘A’
- ‘A’-->‘C’-->‘E’
- ‘F’-->‘D’-->‘B’

IDMLI10

- Consider method F in Java and a singly linked list L below. Suppose that H is the head node of the list L. What is the result if we call F(H.getNext())?

```
public static void F(SLNode node)
{
  if (node!=null)
  {
    System.out.println(node.getData());
    if (node.getNext()!=null)
      F(node.getNext().getNext());
  }
}

L={ 'A' --> 'B' --> 'C' --> 'D' --> 'E' --> 'F' }
H is the head node of L, H= 'A'
```

Select one:

- ‘F’-->‘D’-->‘B’
- ‘E’-->‘C’-->‘A’
- ‘B’-->‘D’-->‘F’
- ‘A’-->‘C’-->‘E’

IDMLI11

– Consider method F in Java and a singly linked list L below. Suppose that H is the head node of the list

L. What is the result if we call F(H.getNext())?

Select one:

- ‘B’-->‘D’-->‘F’
- ‘E’-->‘C’-->‘A’
- ‘A’-->‘C’-->‘E’
- ‘F’-->‘D’-->‘B’

IDMLI12

– What is the number of comparisons needed in the worst case to search for a given node in a Singly Linked List of the length N nodes?

Select one:

- N/2
- log(N)
- N
- Nlog(N)

IDMSOA01

– Selection sort algorithm is used to sort the array A={23,78,45,8,32,56} in the ascending order. What are the items of A after 03 sort pass?

Select one:

- A={8,32,23,45,56,78}
- A={23,32,8,45,56,78}
- A={78,45,56,23,32,8}
- A={78,45,56,8,32,23}

IDMSOA02

– Insertion sort algorithm is used to sort the array A={23,78,45,8,32,56} in the ascending order. What are the items of A after 03 sort pass?

Select one:

- A={45,56,78,32,23,8}
- A={45,56,78,23,32,8}
- A={8,23,45,56,32,78}

- A={8,23,45,78,32,56}

IDMSOA03

– Bubble sort algorithm is used to sort the array A={23,78,45,8,32,56} in the ascending order. What are the items of A after 03 sort pass?

Select one:

- A={8,23,32,45,56,78} (đáp án đúng là {8, 23,32,45,78,56 } cái này gần nhất nên chắc là ok :3)
- A={23,45,8,32,56,78}
- A={8,32,23,45,56,78}
- A={23,32,78,56,45,8}

IDMSOA05

– A sorting algorithm is used to sort the array A={51,11,56,83,20,26,33}. The item of A in each sort pass are listed below. Which sorting algorithm is used?

Select one:

- Selection sort
- Merge sort
- Bubble sort
- Insertion sort

IDMSOA06

– A sorting algorithm is used to sort the array A={83,8,12,72,71,65,5}. The item of A in each sort pass are listed below. Which sorting algorithm is used?

Pass 1:	8	12	72	71	65	5	83
Pass 2:	8	12	71	65	5	72	83
Pass 3:	8	12	65	5	71	72	83
Pass 4:	8	12	5	65	71	72	83
Pass 5:	8	5	12	65	71	72	83
Pass 6:	5	8	12	65	71	72	83
Pass 7:	5	8	12	65	71	72	83

Select one:

- Selection sort
- Merge sort
- Bubble sort
- Insertion sort

IDMSOA08

– Which array represents a Min-Heap?

Select one:

- A={2,5,9,22,10,13,12,8,50}
- A={50,22,13,12,10,8,9,5,2}
- A={2,5,9,8,10,13,12,22,50}
- A={9,5,2,8,10,13,12,22,50}

IDMSOA09

– Which array represents a Max-Heap?

Select one:

- A={78,56,45,32,23,8,15}
- A={8,15,23,32,56,45,78}
- A={78,23,15,56,32,8,45}
- A={8,78,56,32,15,23,45}

IDMSOA10

– An array A contains integer item in the range 0 to 5. A={1,2,5,3,2}. Counting sort algorithm is used to sort A. What is the content of the counting array C before we used the information form C to creates the sorted result array B?

Select one:

- C={1,1,5,5,3,2}.
- C={1,2,5,3,2,5}
- C={0,1,2,1,0,1}
- C={0,1,3,4,4,5}

IDMSOA11

– An array contains integer items, each item has 03 digits. A={170,145,275,900,802}. Radix sort algorithm is used to sort A. What is the content of A after the second sort pass?

Select one:

- A={900,802,145,170,275}
- A={145,170,275,802,900}
- A={900,802,275,170,145}
- A={145,900,170,802,275}.

IDMSOA12

– The Merge method in Merge sort algorithm is used to combine two sorted array A={3,27,38,43} and B={9,10,82}. What is the result array C?

Select one:

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

IDMSOA13

– Heap sort algorithm is used to sort the array A={15,19,10,7,17,16} in the ascending order (using a Max-Heap). What is the content of A after calling BuildHeap() method?

Select one:

- A={19,17,16,7,15,10}
- A={10,15,17,19,7,16}
- A={10,7,15,19,16,17}
- A={19,10,17,7,15,16}

IDMSQ01

– A stack S has 05 character items, S={"A","B","C","D","E"} where "E" is the top of S. What is the content of S if we perform the following list of operations on the stack: push("F") –-> pop() –-> pop() –-> pop() –-> push("D")?

Select one:

- S={"A","B","D","F"}

- S={“A”,“B”,“C”,“D”}
- S={“C”,“D”,“E”,“F”}
- S={“B”,“E”, “F” ,“D”}

IDMSQ02

A queue Q has 05 character items, Q={“A”, “B”, “C”, “D”, “E”} where “E” is the rear and “A” is the front of the queue. What is the content of Q if we perform the following list of operations on the queue: enqueue(“F”)-->dequeue()-->dequeue()-->dequeque()-->enqueue(“D”)?

Select one:

- Q={“C”, “D”, “E”, “F”}
- Q={“A”, “B”, “C”, “D”}
- Q={“D”, “F”, “A”, “B”}
- Q={“D”, “E” ,“F” ,“D”}

IDMSQ03

– A stack S has 05 character items, S={“5”, “4”, “3”, “2”, “1”} where “1” is the top of S. Which operations must be perform to change S into a new state: S={“5”, “4”, “2”, “3”, “1”}?

Select one:

- pop()-->pop()-->pop()-->push(“2”)-->push(“3”)-->push(“1”)
- pop()-->push(“2”)-->pop()-->push(“3”)-->pop()-->push(“1”)
- push(“2”)-->pop()-->push(“3”)-->pop()-->push(“1”)-->pop()
- push(“2”)-->push(“3”)-->push(“1”)-->pop()-->pop()-->pop()

IDMSQ04

– A queue Q has 05 character items, Q={“5”, “4”, “3”, “2”, “1”} where “1” is the front and “5” is the rear of Q. Which operations must be perform to change Q into a new state: Q={“3”, “2”, “1”, “4”, “5”}?

Select one:

- enqueue(“4”)-->enqueue(“5”)-->dequeue()-->dequeue()
- dequeue()-->enqueue(“5”)-->dequeue()-->enqueue(“4”)
- enqueue(“5”)-->enqueue(“4”)-->dequeue()-->dequeue()
- dequeue()-->dequeue()-->enqueue(“5”)-->enqueue(“4”)

IDMSQ05

In method F below, the stack s contains character items. Which is the result if we call method F with the input string text="datastructure"?

```
public static void F(String text)
{
  Stack s=new Stack();
  for (int i=0; i<text.length(); i++)
    s.push(text.charAt(i));
  while (!s.isEmpty())
   System.out.print(s.pop());
}
```

Select one:

- erutcurtsatad
- datastructure
- erutcurtsataderutcurtsatad
- datastructuredatastructure

IDMSQ07

– One difference between a queue and a stack is:

Select one:

- Stacks use two ends of the structure, queues use only one.
- Queues require linked lists, but stacks do not.
- Queues use two ends of the structure; stacks use only one.
- Stacks require linked lists, but queues do not.

IDMSQ08

– In an array-based stack, which operation has time complexity O(N) in the worst-case?

Select one:

- push().
- No operation that has time complexity O(N).
- isEmpty().
- pop().

IDMSQ09

– In an array-based circular queue, which operation has time complexity O(N) in the worst-case?

Select one:

- isFull().
- enqueue().
- No operation that has time complexity O(N).
- dequeue().

IDMSQ10

– Suppose that you are implementing an operation named multiDequeue(int k) on a queue contains integer items. This operation will perform dequeue() k times and return the result of the kth dequeue(). Please complete the code of the operation?

```
public int multiDequeue(int k)
{
    int m=k;
    int result;
    while ((!isEmpty())&&(m>0))
    {
       result=dequeue();
       ----;
    }
    return result;
}
```

Select one:

- k=k-1
- dequeue(k)
- enqueue(k)
- m=m-1

IDMSQ11

– Suppose that you are implementing an operation name multiPop(int k) on a stack contains integer items. This operation will perform pop() k times and return the result of the kth pop().

Please complete the code of the operation?

```
public int multiPop(int k)
{
    int m=k;
    int result;
    while ((!isEmpty())&&(m>0))
    {
       result=pop();
       ----;
    }
    return result;
}
```

Select one:

- k=k-1
- pop(k)
- m=m-1
- push(k)

IDMSQ12

A queue q has 5 items. How many items left in q after executing: q.enqueue(q.dequeue())?

Select one:

- 0
- 4
- 6
- 5

IDMSQ13

– In Java the run-time error “Array index out of bound” is reported when a program try to access an array with an index that is outside the valid boundaries of the array. Which of the following data structure may give the “Array index out of bound” error, even though the current number of elements in it, is less than its size.

Select one:

- Array-based stack.
- Circular array-based queue.
- Simple array-based queue.

- Array-based list.

IDMSQAS01

– Which is the prefix notation of the following infix expression((8+2)*(5+7)-10)*9+3?

Select one:

- +*-*++8 2 5 7 10 9 3
- +*-*+8 2 + 5 7 10 9 3
- 8 2 5 7 10 9 3 *+*-*+
- 8 2 + 5 7 +* 10 -9*3+

IDMSQAS02

– Which is the prefix notation of the following infix expression:5 + (7 + 9*3)*(2+8)?

Select one:

- 5 7 9 3 2 8 * + + * +
- 5 7 9 3 * + 2 8 + * +
- + 5 * + 7 * 9 3+ 2 8
- + * + * + 5 7 9 3 2 8

IDMSQAS03

– Evaluate the following expression:8 7 + 6 4 + * 2 3 7 + * - 1 -?

Select one:

- 129
- 192
- 219
- 291

IDMSQAS04

– Evaluate the following expression:*-*+7 3 + 9 1 * 5 + 2 8 3?

Select one:

- 510
- 501

150

105

IDMSQAS05

– Method F below takes a number n as an argument, and use a stack s to do processing. What does the method do in general?

```
public void F(int n)
{
   Stack s = new Stack();
   while (n>0)
   {
     s.push(n % 2);
     n = n / 2;
   }
   while (!s.isEmpty())
     System.out.print(s.pop());
}
```

Select one:

Print binary representation of n in reverse order.

Print binary representation of n.

Print all positive even number that is smaller than n.

Print all positive even number that is smaller than n in reverse order.

IDMSQAS06

– In the following list of number: 8 17 25 35 41 52 60 75 86. How many comparisons would binary search take to find 52?

Select one:

2

4

6

3

IDMSQAS07

– In the following list of number: 8 17 25 35 41 52 60 75 86. How many comparisons would linear search take to find 52?

Select one:

3

2

4

6

IDMSQAS08

– The keys 12, 18, 13, 2, 3, 23, 15 and 5 are inserted into an initially empty hash table of length 10 using close hashing with hash function: h(k)=k mod 10 and linear probing. What is the

resultant hash table?

0	
1	
2	12
3	3
4	2
5	5
6	15
7	13
8	18
9	23

A

0	
1	
2	12
3	13
4	2
5	3
6	23
7	15
8	18
9	5

B

0	
1	
2	2
3	3
4	
5	5
6	
7	
8	18
9	

C

0	2
1	3
2	12
3	13
4	23
5	5
6	15
7	
8	18
9	

D

Select one:

C

D

B

A

IDMSQAS09

– The keys 12, 18, 13, 2, 3, 23, 15 and 5 are inserted into an initially empty hash table of length 10 using open hashing with hash function: h(k)=k mod 10 and separate chaining. What is the

resultant hash table?

Select one:

- B
- D
- C
- A

IDMSQAS10

– Consider a hash table of size seven, with starting index zero, and a hash function: h(k)=(3k+4) mod 7. Assuming the hash table is initially empty, which of the following is the contents of the table when the sequence 1, 3, 8, 10 is inserted into the table using close hasing with linear probing? Note that '-' denotes an empty location in the table.

Select one:

- 8, -, -, -, -, -,10
- 1, -, -, -, -, -,3
- 1, 10, 8, -, -, -, 3
- 1, 8, 10, -, -, -, 3

IDMSQAS11

– A characteristic of the data that binary search uses but linear search ignores is the?

Select one:

- Maximum and minimum value of the list.
- Type of the list.
- Order of the elements of the list.
- Length of the list.