**Question 1**(1 point)

Which of the following are true about binary and linear search on an array?

Question 1 options:

|  |  |
| --- | --- |
|  | Linear search works by iterating through each element to find a match |
|  | Binary search and linear search are the only two searching algorithms that exist |
|  | Linear search has a time complexity of O(nlog(n)) |
|  | Binary search is the fastest searching algorithm in all cases |

**Question 2**(1 point)

Consider the following Algorithm:

for (i = 0; i < numbersSize - 1; ++i) {

   // Find index of smallest remaining element  
   indexSmallest = i  
   for (j = i + 1; j < numbersSize; ++j) {

      if (numbers[j] < numbers[indexSmallest]) {  
         indexSmallest = j  
      }  
   }

   // Swap numbers[i] and numbers[indexSmallest]  
   temp = numbers[i]  
   numbers[i] = numbers[indexSmallest]  
   numbers[indexSmallest] = temp  
}

This is:

Question 2 options:

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  | | --- | --- | | a) | None of these | |
|  | |  |  | | --- | --- | | b) | Selection Sort | |
|  | |  |  | | --- | --- | | c) | Merge Sort | |
|  | |  |  | | --- | --- | | d) | Insertion Sort | |

**Question 3**(1 point)

Consider the following Algorithm:

for (i = 1; i < numbersSize; ++i) {  
   j = i  
   // Insert numbers[i] into sorted part  
   // stopping once numbers[i] in correct position  
   while (j > 0 && numbers[j] < numbers[j - 1]) {  
         
      // Swap numbers[j] and numbers[j - 1]  
      temp = numbers[j]  
      numbers[j] = numbers[j - 1]  
      numbers[j - 1] = temp  
      --j  
   }  
}

This is:

Question 3 options:

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  | | --- | --- | | a) | None of these | |
|  | |  |  | | --- | --- | | b) | Insertion Sort | |
|  | |  |  | | --- | --- | | c) | Merge Sort | |
|  | |  |  | | --- | --- | | d) | Selection Sort | |

**Question 4**(1 point)

Which of the following sorting algorithms have a time complexity of n2?

Question 4 options:

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  | | --- | --- | | a) | Selection Sort | |
|  | |  |  | | --- | --- | | b) | Insertion Sort | |
|  | |  |  | | --- | --- | | c) | All of these | |
|  | |  |  | | --- | --- | | d) | Bubble Sort | |

**Question 5**(1 point)

Consider the following definition in C programming language.

Struct node {

     Int   data;

     Struct  node  \* next;

}

typedef  struct node NODE;

NODE \*ptr;

Which of the following c code is used to create new node?

Question 5 options:

|  |  |
| --- | --- |
|  | Ptr = (NODE\*)malloc(Sizeof(NODE)); |
|  | Ptr = (NODE\*)malloc(NODE); |
|  | Ptr= (NODE\*)malloc(sizeof(NODE\*)); |
|  | Ptr = (NODE)malloc(sizeof(NODE)); |

**Question 6**(1 point)

In Linked List each node contains a minimum of two fields. One field is the data field to store the data, the second field is?

Question 6 options:

|  |  |
| --- | --- |
|  | Pointer to Integer |
|  | Pointer to character |
|  | Pointer to Node |
|  | Node |

**Question 7**(1 point)

In the Worst Case, the number of comparisons needed to search a single linked list of length n for a given element is ?

Question 7 options:

|  |  |
| --- | --- |
|  | n |
|  | log2n |
|  | log2n - 1 |
|  | n/2 |

**Question 8**(1 point)

What is printed after the following code snippet terminates?

void solve(struct Node \*head) {  
 /\*  The LinkedList is defined as:  
      head-> data = value of the node  
      head-> next = address of next element from the node  
      The List is 1 -> 2 -> 3 -> 4 -> 5    \*/  
   int sum = 0;  
  while(head -> next != NULL) {  
      sum += head -> data;  
      head = head -> next;  
  }  
printf(" %d \n", sum);  
}

Question 8 options:

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  | | --- | --- | | a) | 15 | |
|  | |  |  | | --- | --- | | b) | 20 | |
|  | |  |  | | --- | --- | | c) | 5 | |
|  | |  |  | | --- | --- | | d) | 10 | |
|  | |  |  | | --- | --- | | e) | 1 | |

**Question 9**(1 point)

What is the output of the following code snippet for 1->2->3->4->5?

void solve(struct Node \*head) {  
    while(head != NULL) {  
        printf("%d ", head -> data );  
        head = head -> next;  
    }  
}

Question 9 options:

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  | | --- | --- | | a) | 12345 | |
|  | |  |  | | --- | --- | | b) | 1234 | |
|  | |  |  | | --- | --- | | c) | 54321 | |
|  | |  |  | | --- | --- | | d) | 5432 | |
|  | |  |  | | --- | --- | | e) | 24135 | |

**Question 10**(1 point)

New nodes are added to the \_\_\_\_\_ of the stack.

Question 10 options:

|  |  |
| --- | --- |
|  | Front |
|  | End |
|  | Back |
|  | Top |

**Question 11**(1 point)

**Push(3), push(1), push(2), push(1), pop(), push(3), pop(), pop(),pop(), push(2), pop(),pop().**  
Above operations were applied on a stack, then what is the sequence of popped out values?

Question 11 options:

|  |  |
| --- | --- |
|  | 1,2,3,3,2,1 |
|  | 1,3,2,1,2,3 |
|  | None of these |
|  | 2,3,1,1,3,2 |
|  | 1,2,3,1,2,3 |

**Question 12**(1 point)

int main() {  
    int a=1;  
    int \*p, \*q;  
    p=&a;  
    q=p;  
    printf("%d ", a);  
    \*p=2;  
    printf("%d ", a);  
    \*q=3;  
    printf("%d ", a);  
    return 0;  
}

What is the output of the above code?

Question 12 options:

|  |  |
| --- | --- |
|  | 1 2 3 |
|  | Compile error |
|  | 1 1 1 |
|  | 1 2 2 |

**Question 13**(1 point)

#define MAX 11  
   struct STACK {  
   int arr [MAX];  
   int top = -1;  
}

If the array arr index starts with 0, the maximum value of top which does not cause stack overflow is?

Question 13 options:

|  |  |
| --- | --- |
|  | 11 |
|  | 10 |
|  | 9 |
|  | 8 |

**Question 14**(1 point)

Given numQueue: 1, 55  with 1 at the front.  
Assume Enqueue inserts, what is the result of the following operations:  
  
Enqueue(numQueue, 28)  
Enqueue(numQueue, 72)

Question 14 options:

|  |  |
| --- | --- |
|  | 1, 55, 28, 72 |
|  | 28, 72, 1, 55 |
|  | 72, 28, 1, 55 |
|  | 28, 72 |

**Question 15**(1 point)

What is the time complexity of fun()?

|  |
| --- |
| int fun(int n){    int count = 0;    for (int i = 0; i < n; i++)       for (int j = i; j > 0; j--)          count = count + 1;    return count;  } |

Question 15 options:

|  |  |
| --- | --- |
|  | θ(n2) |
|  | θ(nlog⁡n) |
|  | θ(nlog2⁡n) |
|  | θ(n) |

**Question 16**(1 point)

Which of the following is a correct ordering of complexity?

Question 16 options:

|  |  |
| --- | --- |
|  | 1<n<lgn<n^3<2^(1/2lgn) |
|  | n<n2<n3<n^(n!n)<n! |
|  | lgn<n<n^2<8^lgn<n^n |
|  | n<n^2<n^3<4^lgn<n!   * 밑과 로그를 바꿀수 있다. 8^(lgn) = n^(lg2^3) = n^3 |

**Question 17**(1 point)

Consider a queue containing the values 43, 12, and 77, with 43 at the front of the queue. What value is returned and what is the contents of the queue, after performing the operation **peek(queue)**?

Question 17 options:

|  |  |
| --- | --- |
|  | 77 is returned, and the queue contains 43, 12. |
|  | 77 is returned, and the queue contains 43, 12, 77. |
|  | 43 is returned, and the queue contains 43, 12, 77. |
|  | 4 is returned, and the queue is empty. |
|  | 43 is returned, and the queue contains 12, 77. |

**Question 18**(1 point)

Which of the following is LIFO?

Question 18 options:

|  |  |
| --- | --- |
|  | Stack |
|  | Binary tree |
|  | Queue |
|  | Linked list |

**Question 19**(1 point)

Binary search is ALWAYS faster than linear search.

Question 19 options:

|  |  |
| --- | --- |
|  | True |
|  | False |

**Question 20**(1 point)

Merge sort algorithm makes use of a 'swap' to sort a given array.

Question 20 options:

|  |  |
| --- | --- |
|  | True |
|  | False |