“Singly Linked List Operations – 1”.

1. A linear collection of data elements where the linear node is given by means of pointer is called?
a) Linked list
b) Node list
c) Primitive list
d) Unordered list
View Answer
Answer: a
Explanation: In Linked list each node has its own data and the address of next node. These nodes are linked by using pointers. Node list is an object that consists of a list of all nodes in a document with in a particular selected set of nodes.

2. Consider an implementation of unsorted singly linked list. Suppose it has its representation with a head pointer only. Given the representation, which of the following operation can be implemented in O(1) time?

```
i) Insertion at the front of the linked list
ii) Insertion at the end of the linked list
iii) Deletion of the front node of the linked list
iv) Deletion of the last node of the linked list
```

a) I and II
b) I and III
c) I, II and III
d) I, II and IV
View Answer
Answer: b
Explanation: We know the head node in the given linked list. Insertion and deletion of elements at the front of the linked list completes in O (1) time whereas for insertion and deletion at the last node requires to traverse through every node in the linked list. Suppose there are n elements in a linked list, we need to traverse through each node. Hence time complexity becomes O(n).

3. In linked list each node contains a minimum of two fields. One field is data field to store the data second field is?
a) Pointer to character
b) Pointer to integer
c) Pointer to node
d) Node
View Answer

Answer: c
Explanation: Each node in a linked list contains data and a pointer (reference) to the next node. Second field contains pointer to node.

4. What would be the asymptotic time complexity to add a node at the end of singly linked list, if the pointer is initially pointing to the head of the list?
a) O(1)
b) O(n)
c) θ(n)
d) θ(1)
View Answer
Answer: c
Explanation: In case of a linked list having n elements, we need to travel through every node of the list to add the element at the end of the list. Thus asymptotic time complexity is θ(n).

5. What would be the asymptotic time complexity to insert an element at the front of the linked list (head is known)?
a) O(1)
b) O(n)
c) $O(n^2)$
d) $O(n^3)$
View Answer
Answer: a
Explanation: To add an element at the front of the linked list, we will create a new node which holds the data to be added to the linked list and pointer which points to head position in the linked list. The entire thing happens within O (1) time. Thus the asymptotic time complexity is O (1).

6. What would be the asymptotic time complexity to find an element in the linked list?
a) O(1)
b) O(n)
c) $O(n^2)$
d) $O(n^4)$
View Answer
Answer: b
Explanation: If the required element is in the last position, we need to traverse the entire linked list. This will take O (n) time to search the element.

7. What would be the asymptotic time complexity to insert an element at the second position in the linked list?
a) O(1)
b) O(n)

c) $O(n^2)$
d) $O(n^3)$
View Answer
Answer: a
Explanation: A new node is created with the required element. The pointer of the new node points the node to which the head node of the linked list is also pointing. The head node pointer is changed and it points to the new node which we created earlier. The entire process completes in O (1) time. Thus the asymptotic time complexity to insert an element in the second position of the linked list is O (1).

8. The concatenation of two lists can be performed in O(1) time. Which of the following variation of the linked list can be used?
a) Singly linked list
b) Doubly linked list
c) Circular doubly linked list
d) Array implementation of list
View Answer
Answer: c
Explanation: We can easily concatenate two lists in O (1) time using singly or doubly linked list, provided that we have a pointer to the last node at least one of the lists. But in case of circular doubly linked lists, we will break the link in both the lists and hook them together. Thus circular doubly linked list concatenates two lists in O (1) time.

9. 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?
a) ptr = (NODE*)malloc(sizeof(NODE));
b) ptr = (NODE*)malloc(NODE);
c) ptr = (NODE*)malloc(sizeof(NODE*));
d) ptr = (NODE)malloc(sizeof(NODE));
View Answer
Answer: a
Explanation: As it represents the right way to create a node.

“Singly Linked Lists Operations – 2”.

1. What kind of linked list is best to answer questions like “What is the item at position n?”
a) Singly linked list
b) Doubly linked list
c) Circular linked list
d) Array implementation of linked list
View Answer
Answer: d
Explanation: Arrays provide random access to elements by providing the index value within square brackets. In the linked list, we need to traverse through each element until we reach the nth position. Time taken to access an element represented in arrays is less than the singly, doubly and circular linked lists. Thus, array implementation is used to access the item at the position n.

2. Linked lists are not suitable for the implementation of _____
a) Insertion sort
b) Radix sort
c) Polynomial manipulation
d) Binary search
View Answer
Answer: d
Explanation: It cannot be implemented using linked lists.

3. Linked list is considered as an example of _____ type of memory allocation.
a) Dynamic
b) Static
c) Compile time
d) Heap
View Answer
Answer: a
Explanation: As memory is allocated at the run time.

4. In Linked List implementation, a node carries information regarding _____
a) Data
b) Link
c) Data and Link
d) Node
View Answer

Answer: c
Explanation: A linked list is a collection of objects linked together by references from an object to another object. By convention these objects are names as nodes. Linked list consists of nodes where each node contains one or more data fields and a reference(link) to the next node.

5. Linked list data structure offers considerable saving in _____
a) Computational Time
b) Space Utilization
c) Space Utilization and Computational Time
d) Speed Utilization
View Answer
Answer: c
Explanation: Linked lists saves both space and time.

6. Which of the following points is/are not true about Linked List data structure when it is compared with an array?
a) Arrays have better cache locality that can make them better in terms of performance
b) It is easy to insert and delete elements in Linked List
c) Random access is not allowed in a typical implementation of Linked Lists
d) Access of elements in linked list takes less time than compared to arrays
View Answer
Answer: d
Explanation: To access an element in a linked list, we need to traverse every element until we reach the desired element. This will take more time than arrays as arrays provide random access to its elements.

7. What does the following function do for a given Linked List with first node as head?

```
void fun1(struct node* head)
{
    if(head == NULL)
    return;
    fun1(head->next);
    printf("%d  ", head->data);
}
```

a) Prints all nodes of linked lists
b) Prints all nodes of linked list in reverse order
c) Prints alternate nodes of Linked List
d) Prints alternate nodes in reverse order
View Answer

Answer: b
Explanation: fun1() prints the given Linked List in reverse manner.
For Linked List 1->2->3->4->5, fun1() prints 5->4->3->2->1.

8. Which of the following sorting algorithms can be used to sort a random linked list with minimum time complexity?
a) Insertion Sort
b) Quick Sort
c) Heap Sort
d) Merge Sort
View Answer
Answer: d
Explanation: Both Merge sort and Insertion sort can be used for linked lists. The slow random-access performance of a linked list makes other algorithms (such as quicksort) perform poorly, and others (such as heapsort) completely impossible. Since worst case time complexity of Merge Sort is O(nLogn) and Insertion sort is $O(n^2)$, merge sort is preferred.

"Singly Linked Lists Operations – 3".

1. The following function reverse() is supposed to reverse a singly linked list. There is one line missing at the end of the function.

```
/* Link list node */
struct node
{
    int data;
    struct node* next;
};

/* head_ref is a double pointer which points to head (or start) pointer
  of linked list */
static void reverse(struct node** head_ref)
{
    struct node* prev   = NULL;
    struct node* current = *head_ref;
    struct node* next;
    while (current != NULL)
    {
```

```
        next  = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    /*ADD A STATEMENT HERE*/
}
```

What should be added in place of "/*ADD A STATEMENT HERE*/", so that the function correctly reverses a linked list.
a) *head_ref = prev;
b) *head_ref = current;
c) *head_ref = next;
d) *head_ref = NULL;
View Answer
Answer: a
Explanation: *head_ref = prev; At the end of while loop, the prev pointer points to the last node of original linked list.
We need to change *head_ref so that the head pointer now starts pointing to the last node.

2. What is the output of following function for start pointing to first node of following linked list?

```
1->2->3->4->5->6
void fun(struct node* start)
{
    if(start == NULL)
    return;
    printf("%d  ", start->data);
    if(start->next != NULL )
    fun(start->next->next);
    printf("%d  ", start->data);
}
```

a) 1 4 6 6 4 1
b) 1 3 5 1 3 5
c) 1 2 3 5
d) 1 3 5 5 3 1
View Answer
Answer: d
Explanation: fun() prints alternate nodes of the given Linked List, first from head to end,

and then from end to head.
If Linked List has even number of nodes, then skips the last node.

3. The following C function takes a simply-linked list as an input argument. It modifies the list by moving the last element to the front of the list and returns the modified list. Some part of the code is left blank. Choose the correct alternative to replace the blank line.

```
typedef struct node
{
    int value;
    struct node *next;
}Node;

Node *move_to_front(Node *head)
{
    Node *p, *q;
    if ((head == NULL: || (head->next == NULL))
    return head;
    q = NULL; p = head;
    while (p-> next !=NULL)
    {
        q = p;
        p = p->next;
    }
   ____________________________
  return head;
}
```

a) q = NULL; p->next = head; head = p;
b) q->next = NULL; head = p; p->next = head;
c) head = p; p->next = q; q->next = NULL;
d) q->next = NULL; p->next = head; head = p;
View Answer
Answer: d
Explanation: When while loop completes its execution, node ‘p’ refers to the last node whereas the ‘q’ node refers to the node before ‘p’ in the linked list. q->next=NULL makes q as the last node. p->next=head places p as the first node. the head must be modified to ‘p’ as ‘p’ is the starting node of the list (head=p). Thus the sequence of steps are q->next=NULL, p->next=head, head=p.

4. The following C function takes a single-linked list of integers as a parameter and rearranges the elements of the list. The function is called with the list containing the

integers 1, 2, 3, 4, 5, 6, 7 in the given order. What will be the contents of the list after the function completes execution?

```
struct node
{
    int value;
    struct node *next;
};
void rearrange(struct node *list)
{
    struct node *p, * q;
    int temp;
    if ((!list) || !list->next)
      return;
    p = list;
    q = list->next;
    while(q)
    {
         temp = p->value;
         p->value = q->value;
         q->value = temp;
         p = q->next;
         q = p?p->next:0;
    }
}
```

a) 1, 2, 3, 4, 5, 6, 7
b) 2, 1, 4, 3, 6, 5, 7
c) 1, 3, 2, 5, 4, 7, 6
d) 2, 3, 4, 5, 6, 7, 1

View Answer

Answer: b
Explanation: The function rearrange() exchanges data of every node with its next node. It starts exchanging data from the first node itself.

5. In the worst case, the number of comparisons needed to search a singly linked list of length n for a given element is?
a) log 2 n
b) n/2
c) log 2 n – 1
d) n

View Answer

Answer: d
Explanation: In the worst case, the element to be searched has to be compared with all elements of the linked list.

6. Given pointer to a node X in a singly linked list. Only one pointer is given, pointer to head node is not given, can we delete the node X from given linked list?
a) Possible if X is not last node
b) Possible if size of linked list is even
c) Possible if size of linked list is odd
d) Possible if X is not first node
View Answer
Answer: a
Explanation: Following are simple steps.

```
struct node *temp  = X->next;
X->data  = temp->data;
X->next  = temp->next;
free(temp);
```

7. You are given pointers to first and last nodes of a singly linked list, which of the following operations are dependent on the length of the linked list?
a) Delete the first element
b) Insert a new element as a first element
c) Delete the last element of the list
d) Add a new element at the end of the list
View Answer
Answer: c
Explanation: Deletion of the first element of the list is done in O (1) time by deleting memory and changing the first pointer.
Insertion of an element as a first element can be done in O (1) time. We will create a node that holds data and points to the head of the given linked list. The head pointer was changed to a newly created node.
Deletion of the last element requires a pointer to the previous node of last, which can only be obtained by traversing the list. This requires the length of the linked list.
Adding a new element at the end of the list can be done in O (1) by changing the pointer of the last node to the newly created node and last is changed to a newly created node.

8. In the worst case, the number of comparisons needed to search a singly linked list of length n for a given element is?
a) log2 n
b) $n/2$
c) log2 n – 1

d) n
View Answer
Answer: d
Explanation: The worst-case happens if the required element is at last or the element is absent in the list. For this, we need to compare every element in the linked list. If n elements are there, n comparisons will happen in the worst case.

“Singly Linked List”.

1. Which of the following is not a disadvantage to the usage of array?
a) Fixed size
b) There are chances of wastage of memory space if elements inserted in an array are lesser than the allocated size
c) Insertion based on position
d) Accessing elements at specified positions
View Answer
Answer: d
Explanation: Array elements can be accessed in two steps. First, multiply the size of the data type with the specified position, second, add this value to the base address. Both of these operations can be done in constant time, hence accessing elements at a given index/position is faster.

2. What is the time complexity of inserting at the end in dynamic arrays?
a) O(1)
b) O(n)
c) O(logn)
d) Either O(1) or O(n)
View Answer
Answer: d
Explanation: Depending on whether the array is full or not, the complexity in dynamic array varies. If you try to insert into an array that is not full, then the element is simply stored at the end, this takes O(1) time. If you try to insert into an array which is full, first you will have to allocate an array with double the size of the current array and then copy all the elements into it and finally insert the new element, this takes O(n) time.

3. What is the time complexity to count the number of elements in the linked list?
a) O(1)
b) O(n)
c) O(logn)

d) $O(n^2)$

View Answer

Answer: b

Explanation: To count the number of elements, you have to traverse through the entire list, hence complexity is O(n).

**Note: Join free Sanfoundry classes at Telegram or Youtube**



4. Which of the following performs deletion of the last element in the list? Given below is the Node class.

```
class Node
{
          protected Node next;
          protected Object ele;
          Node(Object e,Node n)
          {
                    ele = e;
                    next = n;
          }
          public void setNext(Node n)
          {
                    next = n;
          }
          public void setEle(Object e)
          {
                    ele = e;
          }
          public Node getNext()
          {
                    return next;
          }
          public Object getEle()
          {
                    return ele;
          }
}
class SLL
{
          Node head;
          int size;
          SLL()
```

```
        {
                size = 0;
        }
}
```

a)



```
public Node removeLast()
{
        if(size == 0)
        return null;
        Node cur;
        Node temp;
        cur = head;
        while(cur.getNext() != null)
        {
                temp = cur;
                cur = cur.getNext();
       }
        temp.setNext(null);
        size--;
        return cur;
}
```

b)

```
public void removeLast()
{
        if(size == 0)
        return null;
        Node cur;
        Node temp;
        cur = head;
        while(cur != null)
        {
                temp = cur;
                cur = cur.getNext();
       }
        temp.setNext(null);
        return cur;
}
```

c)

```
public void removeLast()
{
          if(size == 0)
              return null;
          Node cur;
          Node temp;
          cur = head;
          while(cur != null)
          {
                    cur = cur.getNext();
                    temp = cur;
           }
          temp.setNext(null);
          return cur;
}
```

d)

```
public void removeLast()
{
          if(size == 0)
                    return null;
          Node cur;
          Node temp;
          cur = head;
          while(cur.getNext() != null)
          {
                    cur = cur.getNext();
                    temp = cur;
          }
          temp.setNext(null);
          return cur;
}
```

View Answer

5. What is the functionality of the following code?

```
public void function(Node node)
{
          if(size == 0)
```

```
                head = node;
        else
        {
                Node temp,cur;
                for(cur = head; (temp = cur.getNext())!=null; cur = temp);
                cur.setNext(node);
        }
        size++;
}
```

a) Inserting a node at the beginning of the list
b) Deleting a node at the beginning of the list
c) Inserting a node at the end of the list
d) Deleting a node at the end of the list

View Answer

Answer: c
Explanation: The for loop traverses through the list and then inserts a new node as cur.setNext(node);

6. What is the space complexity for deleting a linked list?
a) O(1)
b) O(n)
c) Either O(1) or O(n)
d) O(logn)

View Answer

Answer: a
Explanation: You need a temp variable to keep track of current node, hence the space complexity is O(1).

7. How would you delete a node in the singly linked list? The position to be deleted is given.
a)

```
public void delete(int pos)
{
        if(pos < 0)
        pos = 0;
        if(pos > size)
        pos = size;
        if( size == 0)
        return;
        if(pos == 0)
        head = head.getNext();
```

```
        else
        {
            Node temp = head;
            for(int i=1; i<pos; i++)
           {
                  temp = temp.getNext();
           }
            temp.setNext(temp.getNext().getNext());
        }
            size--;
}
```

b)

```
public void delete(int pos)
{
        if(pos < 0)
        pos = 0;
        if(pos > size)
        pos = size;
        if( size == 0)
        return;
        if(pos == 0)
        head = head.getNext();
        else
        {
            Node temp = head;
            for(int i=1; i<pos; i++)
            {
                  temp = temp.getNext();
            }
            temp.setNext(temp.getNext());
        }
            size--;
}
```

c)

```
public void delete(int pos)
{
       if(pos < 0)
         pos = 0;
         if(pos > size)
```

```
        pos = size;
        if( size == 0)
        return;
        if(pos == 0)
        head = head.getNext();
        else
        {
            Node temp = head;
            for(int i=1; i<pos; i++)
            {
                  temp = temp.getNext().getNext();
           }
            temp.setNext(temp.getNext().getNext());
        }
            size--;
}
```

d)

```
public void delete(int pos)
{
        if(pos < 0)
        pos = 0;
        if(pos > size)
        pos = size;
        if( size == 0)
         return;
         if(pos == 0)
         head = head.getNext();
         else
         {
             Node temp = head;
             for(int i=0; i<pos; i++)
             {
                  temp = temp.getNext();
             }
             temp.setNext(temp.getNext().getNext());
         }
         size--;
}
```

View Answer

Answer: a
Explanation: Loop through the list to get into position one behind the actual position given. temp.setNext(temp.getNext().getNext()) will delete the specified node.

8. Which of these is not an application of a linked list?
a) To implement file systems
b) For separate chaining in hash-tables
c) To implement non-binary trees
d) Random Access of elements
View Answer
Answer: d
Explanation: To implement file system, for separate chaining in hash-tables and to implement non-binary trees linked lists are used. Elements are accessed sequentially in linked list. Random access of elements is not an applications of linked list.

9. Which of the following piece of code has the functionality of counting the number of elements in the list?
a)

```
public int length(Node head)
{
          int size = 0;
          Node cur = head;
          while(cur!=null)
          {
              size++;
              cur = cur.getNext();
          }
          return size;
}
```

b)

```
public int length(Node head)
{
         int size = 0;
          Node cur = head;
          while(cur!=null)
          {
              cur = cur.getNext();
              size++;
```

```
        }
        return size;
}
```

c)

```
public int length(Node head)
{
        int size = 0;
        Node cur = head;
        while(cur!=null)
        {
            size++;
            cur = cur.getNext();
        }
}
```

d)

```
public int length(Node head)
{
        int size = 0;
        Node cur = head;
        while(cur!=null)
        {
            size++;
            cur = cur.getNext().getNext();
        }
        return size;
}
```

View Answer

Answer: a

Explanation: 'cur' pointer traverses through list and increments the size variable until the end of list is reached.

10. How do you insert an element at the beginning of the list?

a)

```
public void insertBegin(Node node)
{
        node.setNext(head);
```

```
        head = node;
        size++;
}
```

b)

```
public void insertBegin(Node node)
{
        head = node;
        node.setNext(head);
        size++;
}
```

c)

```
public void insertBegin(Node node)
{
        Node temp = head.getNext()
        node.setNext(temp);
        head = node;
        size++;
}
```

d)

```
public void insertBegin(Node node)
{
        Node temp = head.getNext()
        node.setNext(temp);
        node = head;
        size++;
}
```

View Answer

Answer: a

Explanation: Set the 'next' pointer point to the head of the list and then make this new node as the head.

11. What is the functionality of the following piece of code?

```
public int function(int data)
{
```

```
        Node temp = head;
        int var = 0;
        while(temp != null)
        {
                if(temp.getData() == data)
                {
                        return var;
                }
                var = var+1;
                temp = temp.getNext();
        }
        return Integer.MIN_VALUE;
}
```

a) Find and delete a given element in the list
b) Find and return the given element in the list
c) Find and return the position of the given element in the list
d) Find and insert a new element in the list

View Answer

Answer: c

Explanation: When temp is equal to data, the position of data is returned.