Question 1

In a given singly linked list, a pointer is pointing to node x in the list. There is no head pointer in this linked list. We need to perform a delete operation on a random node using pointer x. When does the delete operation fail?

Question 2

A linked list L is given below and a pseudo code is written in which the last node of the linked list is moved to the front of the list. Choose the order of the statements for the operation to be performed successfully from the given three statements below.

I. L = P
II. Q->next = NULL
III. P->next = L

Given linked list L: A -> B -> C -> D -> NULL

```
P = L
while (P->next != NULL)
{
     Q = P
      P = P->next
}

S1: ______________
S2: ______________
S3: ______________
```

Question 3

```
typedef struct node
{
    int data;
    node* next ;
} node;

void join(node* m, node* n)
{
    node* p = n;
    while (p->next != NULL)
    {
        p = p->next;
    }
    p–>next = m;
}
```

Assuming that m and n point to valid NULL-terminated linked lists, invocation of join will ……………

Question 4

In a singly linked list, what does the last node point to?

Question 5

Write a statement to allocate space for a new node of type ‘Node’ for a linked list.

Question 6

Given below is a C code snippet to reverse a singly linked list. Fill in the blank.

```
void reverse(Node* head)
   {
      // Initialize current, previous and next pointers
      Node* current = head;
      Node *prev = NULL, *nxt = NULL;

      while (current != NULL) {

         S1:

         S2:

         prev = current;
         current = nxt;
      }
      head = prev;
   }
```

Question 7

Given below is a C code snippet to pairwise swap the elements of a singly linked list L
Original Linked List **L : 1->2->3->4->5->6->NULL**
After pairwise swapping **L: 2->1->4->3->6->5->NULL**

```
Node* pairWiseSwap(struct Node* head)
{
    struct Node* temp = head;

    while (____________)
   {
        /* Swap data of node with its next node's data */
        swap(&temp->data, &temp->next->data);

        /* Move temp by 2 for the next pair */
        temp = temp->next->next;
    }
    return head;
}
```

What should be the while loop condition for the function to perform pairwise swapping?

Question 8

Consider the C code snippet below.

```
void func(struct Node** head_ref) {
    if (*head_ref == NULL || (*head_ref)->next == NULL) {
        return;
    }

    struct Node* prev = *head_ref;
    struct Node* curr = (*head_ref)->next;
    struct Node* temp;

    while (curr != NULL && prev != NULL) {
        temp = curr;
        prev->next = curr->next;
        free(temp);
        prev = prev->next;
        if (prev != NULL) {
            curr = prev->next;
        }
    }
}
```

What does the given function func do?

Question 9

Consider the following C code snippet used to detect a loop in a linked list

```
P = head;
Q = head->next

while(A)
{
    if (p == q) break; //loop detected

   P = P -> next;
   Q = (Q -> next)? (Q -> next -> next) : (Q -> next);
}
```

What should be placed in place of A to fill the while loop condition?

Question 10

We have a linked list with elements 1, 2, 3, 4, 5, 6, 7, and 8 inserted sequentially to the list, such that the head of the linked list is element 1.

```
struct node
{
  int data;
  struct node *next;
};

void print(struct node* ptr, int pos)
{
    if(____S1_____)
        return;
    if(____S2_____)
        printf("%d", ptr->data);

   ______S3______;

  if(_____S4_____)
      printf("%d", ptr->data);

}
```

The initial call from the main function looks like print(head, 0), where head is a pointer to the head of the linked list. If you want the print function to print the sequence 1 3 5 7 8 6 4 2, then fill in the missing statements S1 to S4?

Question 11

We have a structure defined as:

```
struct node
{
  int data;
  struct node *next;
};
```

We implement a linked list using the above structure definition. To the list we insert values 1,2,3,4,5,6,7,8 sequentially. So the head points to the node containing value 1. Consider the given C code snippet below

```
void modify()
{
    struct node* ptr;
    ptr = head;
    while(ptr->next != NULL)
    {
        ptr = ptr->next;
    }

    ptr->next = head->next;
    head->next = ptr->next->next;
    ptr->next->next = head->next->next->next;
    head->next->next->next = ptr->next->next;
    ptr->next->next = NULL;
}


void print()
{
    struct node* ptr = head;
    while(ptr != NULL)
    {
       printf("%d", ptr->data);
        ptr = ptr->next;
    }

}
```

After executing the modify() function, if we want to print the elements of the linked list using the print() function, the the output will be: …………………

Question 12

Consider the C code snippet below:

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* next;
};

void func(struct Node** head_ref, int position, int data) {
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
    new_node->data = data;
    if (position == 1) {
        new_node->next = *head_ref;
        *head_ref = new_node;
        return;
    }
    struct Node* temp = *head_ref;
    for (int i = 1; i < position - 1 && temp != NULL; i++) {
        temp = temp->next;
    }
    if (temp == NULL) {
        printf("Invalid position.\n");
        return;
    }
    new_node->next = temp->next;
    temp->next = new_node;
}

// Function to print the linked list
void printList(struct Node* head) {
    while (head != NULL) {
        printf("%d ", head->data);
        head = head->next;
    }
    printf("\n");
}

// Example usage:
int main() {
    struct Node* head = NULL;
    func(&head, 1, 1);
    func(&head, 2, 2);
    func(&head, 3, 4);

    printf("Original linked list: ");
    printList(head);

    return 0;
}
```

What will be the output of the given code snippet?

Question 13

Consider the following function that takes reference to head of a Doubly Linked List as parameter. Assume that a node of doubly linked list has previous pointer as prev and next pointer as next.

```
void fun(struct node **head_ref)
{
    struct node *temp  = NULL;
    struct node *current = *head_ref;

   while(current != NULL)
   {
     temp = current->prev;
     current->prev = current->next;
     current->next = temp;
     current = current->prev;
   }

   if(temp != NULL)
   {
      *head_ref = temp->prev;
   }

}
```

Assume that reference of head of following doubly linked list is passed to above function 6 <--> 4 <--> 2 <--> 5 <--> 3 <-->1. What should be the modified linked list after the function call?

Question 14

How many pointers' values need to be modified to insert an element into a doubly linked list?

Question 15

What does the following routine accomplish in a doubly linked list? Let M represent the address of a middle node and N represent the address of a newly inserted node. The routine performs the following actions:

- Assigns the left pointer of N to M.
- Assigns the right pointer of N to the right pointer of M.
- Assigns the right pointer of M to N.
- Assigns the left pointer of the node pointed to by the right pointer of M to N.

At which positions, does the routine insert the new node N?

Question 16

Inserting an element in the middle of a singly linked list requires the modification of how many pointers?

Question 17

What does the function fun() do?

```
int fun(struct Node* head)
{

    if (head == NULL)
        return 1;
    struct Node* ptr;
    ptr = head->next;

    while (ptr != NULL && ptr != head)
        ptr = ptr->next;
    // Condition for circular linked list
    return (ptr == head);
}
```

Question 18

A doubly linked list is one in which

I. each node is linked to the node before it and after it
II. one of the pointers in the first node points to NULL
III. cannot be traversed in the backward direction
IV. has double the number of pointers than a singly linked list

Which of the above statements are correct for a doubly linked list?

Question 19

A circular linked list is one in which

I. each node is linked to the node before it and after it
II. the last node's next pointer has the address of the first node
III. the last node's next pointer points to the second last node
IV. has double the number of pointers than a singly linked list

Which of the above statements is correct for a circular linked list?

Question 20

The following C function takes a singly 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;
    }
}
```