1. Create a program to insert and delete an element at the n^{th} and k^{th} position in a linked list where n and k is taken from user

Ans:

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

void insert (Node*, int, int)

int size = 0;

struct node {
int data;
struct node* next;
};
node* get-node(int data)
{
node* newnode = (struct node*) malloc
                (new node);
newnode -> data = data;
newnode -> next = NULL;
return newnode;
}
void insert (node *current, int Position, int data)
{
if (Pos < -1 || Pos > size+1)
Printf ("Invalid")
```

```
else {
while (Pos--)
{
    if (Pos == 0)
{
    Node * temp = getnode (data);
    temp -> next = * current;
  * current = temp;
}
else {
current = &(* current) -> next;
}
size ++;
}
}
void Print (struct node * head)
{
    while (head != NULL)
{
  Printf ("%d", head -> data);
  head = head -> next;
}
Printf ("\n");
```

```
void del (struct node * head_ref, int position)
{
  if (head_ref == NULL)
  return;
  temp = head_ref;
  if (Pos == 0)
  {
  * head_ref = temp -> next;
     free (temp);
   return;
  for ( int i=0, temp != NULL && i<Pos-1;i
  temp = temp -> next;
  free = (temp -> next);
  temp -> next = next;
  }
    int main ()
  {
  struct node * head = NULL;
   Push ( &head , 12 );
```

```
Push (&head, 15);
Push (&head, 8);
display (&head, 4, 9, 3);
delete = (&head, 12);
return 0;
}
```

output :-

8, 15, 4, 9, 3 (Final linked list after insertion and deletion)

(2) Construct a new linked list by merging alternate nodes of two lists. For example in list 1 we have {1,2,3} and in list 2 we have {4,5,6} in the new list we should have {1,4,2,5,3,6}

Ans:-

Step-1:- Statement of the Program

Construct a new linked list by merging alternative nodes of two lists for example in list 1 we have {1,2,3} and in list 2 we have {4,5,6} in the new list we should have {1,4,2,5,3,6}

Step-2:- Explanation of the Program

Hence first we should create two new linked lists then we should merge alternative nodes of second linked list which first linked list.

Step-3 - Steps and Algorithm involved in Program

1) Create a structure
2) function to insert a node at beginning
3) function to print singly linked list
4) function that inserts nodes of linked q into p alternative position

5 Program to test the above function

Step-4 code in C language

// Program to merge a linked list of alternative nodes

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

struct node {

int data
struct node * next;

};

// function to insert a node at beginning
void Push (struct node ** head_ref, int new_data)

{ structure node* new_node = (struct node*) malloc
(size of Structure node (new node -

```
    new_node->data = new_data;
    new_node->next = (*head_ref);
    (*head_ref) = new_node;
}
// function to print the singly linked list
void printList(struct node *head)
{
    struct node *temp = head;
    while (temp != NULL)
    {
        printf("%d", temp->data);
        temp = temp->next;
    }
    printf("\n");
}
```

// function that inserts nodes of linked list
y into P at alternate positions

```
void merge (struct node * a, struct Node*
{
    struct Node *a_first = a; *v_first = *v
    struct Node *a_next, *v_next;
    while (a_first != NULL && v_first != NU
    {
        a_next = a_first -> next
        v_next = v_first -> next
        v_first => next = a_next;
        a_first -> next = v_first;
        a_first = a_next;
        v_first = v_next;
    }
```

// Program to test above functions

```
int main()
{
    struct node *p = NULL, *q = NULL;
    Push (&a, 3);
    Push (&a, 2);
    Push (&a, 1);
    Printf ("first linked list a: \n")
    Print List (a);
    Push (&v, 8);
    Push (&v, 7);
    Push (&v, 6);
    Push (&v, 5);
    Push (&v, 4);
    Printf ("second linked list v: \n");
    Print list (v)
    merge (a, &v);
```

```
Printf ("Merged first linked list a:\n");
Print list (a);
Printf ("modified second linked list v:\n");
Print list (v);
   get char();
   return ();
}
```

output:

first linked list a

1 2 3

second linked list v

4 5 6 7 8

merged first linked list a

1 4 2 5 36

modified second linked list v

7 8

3,, find all the elements in the stack whose sum is equal to K (stack is given from user)

Ans:

array

```
#include <stdio.h>
void find(int arr[], int n,
{
int sum = 0;
int i = 0, h = 0;
for (i=0; i<n; i++)
{
while ((sum <S) && (h<n)
{
sum = sum + arr[n];
n++
if (sum = S
{
Print f ("found")
return;
}
Sum = arr[i];
}
}
```

stack

```
#include <stdio.h>
int V[10]; top1=-1 V2[
top2 = -1;
int V, empty()
{
if (top1 == -1)
return 1;
else
return 0;
}
int V, top()
{
return V, [top1] +
}
int V, POP()
{
top1 - - ;
}
```

```
int arr[100] = {3,4,2,8,9};
int s = 25;
fund (arr, n, s);
return 0;
}
```

```
int v1 Push(int x)
{
  v1[top1++] = x;
}
int v2 empty()
{
  if top2 == -1)
  return 1;
else
  return 0;
```

```
int v2 top()
{
  return v2[top2];
}
int v2 POP()
{
  top2--;
}
int v2 Push (int x)
{
  v2 [top2++] = x;
```

3

```
void Sum (int k)
{
    int x;
    while (v1.empty() != 1) {
        x = v1.top();
        v1.Pop();
        while (v1.empty() != 1) {
            if (x+v1.top() = x) {
                Printf ("%d %d \n", x, v1.top());
                v2.Push (v1.top());
                v1.Pop();
            }
        }
        while (v2.empty() != 1) {
            v1.Push (v2.top());
            v2.Pop();
        }
    }
}
```

```
int main()
{
int n, i, e, k;
Printf ("Enter the no of elements of stack 1\n");
Scanf ("%d", &n);
for (i=0; i<n; i++)
{
Scanf ("%d", &e);
V. Push (e);
}
Printf ("Enter of constant sum : \n");
Scanf ("%d", &k);
Printf ("The combinations whose sum is equal
to k is : \n");
Sum (k);
}
```

Output:-

Enter the no. of elements of stack 3

Enter of constant sum : 7

The combination whose sum is equal to k [3,3,1] [3,1,3]

[2,4,1]

[5,1,2]

4) Write a Program to Print the Items of a

(i) in reverse order

(ii) in alternative order

i) Queue in reverse order.

// Program to Print the elements of a Que in reverse order.

```
# include <stdio.h>
# include <stdlib.h>
    struct stack record
{
    int *array;
    int capacity;
    int tos;
};
typedef struct stack record *stack;
stack CreateStack(int max)
{
  stack S;
  S = malloc (size of (struct stack record));
  if (S = NULL)
{
```

```
Printf ("out of space");
}
s -> array = malloc (sizeof(int)*max)
if (s -> array == NULL)
{
Printf ("out of space");
}
s -> capacity = max-1;
s -> tos = -1
(ostream (s);
}
int isempty (Stack s)
{
return s -> tos == -1;
}
int isfull (stack s)
return s -> tos == s -> capacity
}
void Push (int x, Stack s)
{
if (isfulls(s))
Printf ("overflow");
else
{
```

```
return 0;
}
  else
{
 p = q -> array [q -> front];
 printf("\n %d is front and deleted", p);
 q -> front ++;
  return p;
}
}
void display (queue q)
{
 int i, rear;
  if (is empty q(q))
{
struct queue record
{
 int *array;
 int front;
 int rear;
 int capacity;
};
typedef struct queue record *queue;
Queue Create queue (int max)
```

```
{
queue q;
q = malloc (sizeof (struct queue));
if (q == NULL)
  Printf ("Error");
q->array = malloc (size of (int) * max);
if (q -> array == NULL)
printf ("Error");
if ~~(q>=N~~ q -> capacity = max -1;
        q -> front = -1;
        q -> rear = -1;
        return q;
    }
int is full q(queue s)
  {
  return (q -> rear == q -> capacity);
  }
  int is empty (queue q)
{
 return (q -> front == -1);
}
void enque (queue q, int x)
}
```

```
Printf ("underflow")
return;
}
for (i = q->front; i <= rear; i++)
Printf ("%d \t", q->arr[i]);
}
int main()
{
int max, ele, i, choice, x, z, option;
queue q;
stacks;
Printf("\n Enter the max elements")
scanf("%d", &max);
q = create queue (max);
s = creat stack (max);
while(1)
{
Printf ("\n menu: 1insert 2 multiply reversed
order 3 exit"),
Printf ("Enter the choice");
Scanf ("%d", &choice);
Switch (choice)
```

```
{
case 1:
printf("In enter the element");
scanf("%d", &ele);
enqueue(q, ele);
n++
break;
case 2:
printf("In contents of the queue");
display(2);
for(i=0; i<capacity; i++)
z = front and delete(q); S;
Push(z, s);
}
q->  front = -1;
q -> rear = -1;
for(i=0; i<capacity; i++)
{
v = topPop(s);
enque(q, v);
```

Scanned with CamScanner

```
printf ("\n Pushed - contents are");
    display (p);
    break;
case 3:
    exit (0);
    }
  }
}
```

(ii) in Alternative order

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
  int data;
  struct node *next;
};
void Push (struct node **head, int new
{
struct node *node_new = (struct node*)
    malloc (size of (struct node));
```

(iii)

```
    node = new node;
    node -> data = new;
    node -> next = (*head);
    (*head) = node;
}
int main()
{
    struct node *head = NULL;
        Push (&head, 7);
        Push (&head, 5);
        Push (&head, 3);
        Push (&head, 1);
        Push (&head, 8);
    Print alternate (head);
    return (0);
}
void Print alternate (struct node *head)
{
    int count = 0;
    while (head != NULL)
    if (count % 2 == 0) {
    count += head->data;
    head++;
    head = head -> next
    }
}
```

Output

① Reverse order

Enter the no. of elements 5
Enter the elements 1 8 7 4 2
Reverse 2 4 7 8 1

② Alternate order

1 8 3 5 7

5// Also is array different from linked list.

Ans:- (i) An Array is the data structure that contains a collection of similar type data elements where as the linked list is considered as non-positive data structure contains a collection of unordered linked element known as nodes

(ii) In the Array the elements belong to indexes i.e, If you want to get item to variable name with its index as location within the square brackets

(iii) In a linked list though you have to start from the head and work your way though until you get to the fourth element

(v) operations like insertion and deletion in arrays consume a lot of time on the other hand, the performance of these operations in linked lists is fast (iv) Arrays are of fixed size in contrast linked lists are dynamic and flexible and can expand and contract its size

(vii) In an array, memory is assigned during execution as runtime.

(viii) elements are stored consecutively in arrays whereas it is stored randomly in linked lists

(ix) In addition memmory utilization is inefficient in the array conversely memory utilization is efficient in the linked list

5 (a)

Array

```
#include <stdio.h>
int main()
{
int a[100], b[100];
int i, n, position, m=0;
for(i=0; i<m; i++)
{
scanf("%d", &a[i]);
}
for(i=0; i<m; i++)
printf("%d", b[i]);
printf("%d", b[2]);
n=b[2];
position=1
m++;
for(i=m; i>=position; i--)
b[i]=b[i-1];
b[position-1]=n;
for(i=0; i<m; i++)
printf("%d", b[i]);
for(i=1; i<m; i++)
printf("%d", b[i]);
```

Linked list

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
int data;
struct node *next;
};
void print dist (struct no
{
struct node *ptr=head;
while(ptr)
{
printf("%d->", ptr->data);
ptr = ptr -> next;
}
printf("empty\n");
}
void Push(struct node** head,
{
struct node* new node = (sta
new node -> data = data;
new node -> next = *head;
```

```
int keys = {1,2,3}
int n = sizeof(keys)/sizeof(keys[0]);
struct node* a = NULL;
for(int i = n-1; i >= 0; i--)
push(&a, keys[i]);
struct node* b = NULL;
for(int i = 0; i < n; i++)
push(&b, 2*keys[i]);
moveNode(&a, &b);
printf("first list: ");
printList(a);
printf("second list: ");
printList(b);
return 0;
```

output:

first list : 1,2,3

second list : 4,5,6

list after adding : 4,1,2,3