① write 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.

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
  struct node *next;
  int n;
};
struct node *curr, *temp, *prev, *last;
struct node *create (struct node *);
void insend (struct node *);
void delpos (struct node *);
void inspos (struct node *);
void display (struct node *);
void main()
{
    struct node *s;
    int k, n, i, j, ch;
    s = NULL;
    do
    {
       printf (" Enter the choice : ");
```

```
printf(" \n * choice: \n 1. create \n 2. continue \n
        3. display \n 4. Insertpos \n 5. delpos \n 6.
        Exit \n");
scanf(" %d ", &ch)
switch (ch)
{
    case 1 : { s = create(s); break;
             }
    case 2 : { ins2nd(s); break;
             }
    case 3 : { display(s); break;
             }
    case 4 : { Inspos(s); break;
             }
    case 5 : { delpos(s); break;
             }
    case 6 : { exit(0);
             }
    default : {
              printf(" Wrong - choice \n ");
             }
}
} while (ch! = 6)
```

```
}
struct node * create (struct node *x)
{
        if (x == NULL)
        {
            x = (struct node *) malloc (sizeof (struct node));
            printf ("Enter the Number\n");
            scanf ("%d", &x->n);
            x->next = NULL;
            return x;
        }
       else
       {
           printf (" Already Node was created\n");
           return x;
        }
}
void display (struct node *x)
{
       curr = x;
       while (curr != NULL)
       {
            printf (" %d  ", curr->n);
            curr = curr->next;
       }
}
```

```
void ins2nd(struct node *x)
{
    curr = x;
    while (curr != NULL)
    {
        curr = curr -> next;
    }
    temp = (struct node *) malloc (sizeof (struct node));
    printf (" Enter the number \n");
    scanf ("%d", &temp -> n);
    temp -> next = NULL;
    curr -> next = temp;
}
void inspos (struct node *x)
{
    int pos, c = 1;
    curr = x;
    printf (" Enter the position to be inserted \n");
    scanf ("%d", &pos);
    while (curr -> next != NULL)
    {
        c++;
        if (c = pos)
        {
            temp = (struct node *) malloc (sizeof (struct node));
```

```
            printf("Enter the Number\n");
            scanf("%d", &temp->n);
            temp->next = curr->next;
            curr->next = temp;
            break;
        }
        curr = curr->next;
    }
}
void delpos(struct node *x)
{
    int pos, c=1;
    curr = x;
    printf("Enter the position to be deleted\n");
    scanf("%d", &pos);
    while(curr->next != NULL)
    {
        c++;
        if(c = pos)
        {
            temp = curr->next;
            curr->next = curr->next->next;
            free(temp);
        }
        curr = curr->next;
    }
}
```

Output:

Enter the choice:

1. create
2. continue
3. display
4. Inspos
5. delpos
6. Exit

1

Enter the Number : 15

Enter the choice:

1. create
2. continue
3. display
4. Inspos
5. delpos
6. Exit

2

Enter the Number : 20

Enter the choice :

1. create
2. continue
3. display
4. Inspos
5. delpos
6. Exit

2

Enter the Number : 30

Enter the choice :

1. create
2. continue
3. display
4. Inspos
5. delpos
6. Exit

4

Enter the position to be Inserted : 3

Enter the Number : 25

Enter the choice :

1. create
2. continue
3. display

4. inspos

5. delpos

6. Exit

3

15 20 25 30

Enter the choice :

1. Create

2. continue

3. display

4. Inspos

5 delpos

6. Exit

6

(2) construct a New linked list by merging alternating 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}.

```
#include <stdio.h>
#include <stdlib.h>
struct node {
        int data;
        struct node *link;
} *head1 = NULL, *temp, *temp1, *head2 = NULL,
  *head3 = NULL;
struct node *insert (struct node *head, int x)
{
        temp = (struct node *) malloc (sizeof (struct node));
        temp -> data = x;
       temp -> link = NULL;
       if (head == NULL)
       {
               head = temp;
       }
```

```
        else
        {
                temp1 = head;
                while (temp1 -> link != NULL)
                {
                        temp1 = temp1 -> link;
                }
                temp -> link = temp;
        }
        return head;
}
void main()
{
        int p, q, x, i;
        printf(" Enter the No. of first Linked List \n");
        scanf("%d", &p);
        for(i=0; i<p; i++)
        {
            printf(" Enter the element \n");
            scanf("%d", &x);
            head1 = insert(head1, x);
        }
```

```
printf (" Enter No of elements of second linked list \n");
scanf ("%d", &q)
for(i=0; i<q ,i++)
{
    printf (" Enter the Element \n");
    scanf (" %d", &x);
    head2 = insert (head2, x);
}
temp = head1;
temp1 = head2;
while ( temp!=NULL && temp1 != NULL)
{
    printf (" %d ", temp→data);
    printf ("%d ", temp1→data);
    temp = temp→link;
    temp1 = temp→link;
}
while (temp != NULL)
{
    printf (" %d ", temp→data);
    temp = temp→link;
}
while (temp1 != NULL)
{
    printf (" %d ", temp1→data);
    temp1 = temp1→link;
```

```
    }
}
```

Output:

```
Enter the No of first linked list : 3
Enter the Element : 1
Enter the Element : 2
Enter the Element : 3
Enter the No of second linked list :
Enter the Element : 4
Enter the Element : 5
Enter the Element : 6

1 4 2 5 3 6

1 2 3

4 5 6
```

③ Find all the elements in the stack whose sum is equal to k (where k is given from user).

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

int s1[10], top1 = -1, s2[10], top2 = -1;
int s1 smpty ( )
{
        if (top1 == -1)
                return 1;
        else
                return 0;
}
int s1 top ( )
{
        return s1[top1];
}
int s1 pop( )
{
        top--;
}
int s1 push (int x)
{
        if (top2 == -1)
                return 1;
        else
                return 0;
```

```
}
int s2top()
{
    return s2[top2];
}
int s2pop()
{
    top2--;
}
int s2push(int x)
{
    s2[++top2]=x;
}
int sum(int k)
{
    int x;
    while(s1empty()!=1)
    {
        x=s1top();
        s1pop();
        while(s1empty()!=1)
        {
            if(x+s1top()=k)
            {
                printf("(%d %d)\n",x,s1top());
            }
```

```
                S2 push (S1 top());
                S1 pop();
            }
            while (S2 smpty() != 1)
            {
                S1 push (S2 top());
                S2 pop();
            }
        }
}
int main()
{
    int n, i, e, k;
    printf("Enter the No of elements in stack: \n");
    scanf ("%d", &n);
    for(i=0; i<n; i++)
    {
        scanf("%d", &e);
        S1 push(e);
    }
    printf("Enter the value 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: 5

1

2

3

4

5

Enter the value of constant are sum: 6

The combinations whose sum is equal to K is :

(5, 1)

(4, 2)

④ write a program to print the elements in a Queue.

① in Reverse order

⑪ in Alternate order.

→

① In Reverse order.

```
#include <stdio.h>
int q[20], front=-1, rear=-1;
int s[10], top=-1
int qpush (int x)
{
    if (front==-1)
    {
        q[++rear]=x;
        front++;
    }
    else
    {
        q[++rear]=x;
    }
}
int qfront()
{
    return q[front];
}
int qpop()
{
    front++;
}
```

```
int qsempty( )
{
      if (front > rear)
            return 1;
      else
            return 0;
}
int spop( )
{
     top--;
}
int spush( int x)
{
    s[++top] = x;
}
int sempty( )
{
      if (top==-1)
            return 1;
      else
            return 0;
}
int stop( )
{
       return s[top];
}
```

```
int main()
{
    int n, i, x;
    printf("Enter the No of elements in a Queue");
    scanf("%d", &n);
    for (i=0; i<n; i++)
    {
        scanf("%d", &x);
        qpush(x);
    }
    while (qempty() != 1)
    {
        spush(qfront());
        qpop();
    }
    while (sempty() != 1)
    {
        printf("%d ", stop());
        spop();
    }
}
```

(11) In Alternative order:

```
#include <stdio.h>
#include <stdlib.h>
int q[20], front = -1, rear = -1;
int qpush(int x)
{
        if (front == -1)
        {
                q[++rear] = x;
                front++;
        }
        else
        {
                q[++rear] = x;
        }
}
int qfront()
{
      return q[front];
}
int qpop()
{
      front++;
}
```

```
int qempty()
{
        if (front > rear)
                return 1;
        else
                return 0;
}
int main()
{
        int n, i, x;
        printf ("Enter the No of Elements of Queue : ");
        scanf ("%d", &n);
        for (i=0; i<n; i++)
        {
                scanf ("%d", &x);
                qpush(x);
        }
        i=0;
        while (qempty() != 1)
        {
                if (i%2==0)
                        printf ("%d ", qfront());
                i++;
                qpop();
        }
}
```

Output :

④

① Enter the NO of elements in Queue : 5

9
7
6
3
2

2 3 6 7 9

⑪ Enter the NO of elements in Queue : 5

9
7
6
3
2

9 6 2

⑤

① How array is different from the linked list.

→ Array and linked list are used to store data of similar types.

→ In Array we work with the index; whereas in linked list we start from Head (First) and works by linking the Next nodes.

→

Parameters	ARRAY	LINKED LIST
Size	Fixed	Variable
Memory Allocation	continous memory allocation	Random memory Allocation
Access	Direct Access	sequential Access
Memory utilization	inefficient	efficient
Searching	Binary search, Linear search ..etc.	Linear search

→ Insertion and deletion operations are easy in linked list; whereas in array's it impossible

(11) write a program to add the first element of list 1 to another list

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int data;
    struct node *link;
} *head1 = NULL, *temp, *temp1, *head2 = NULL;
struct node * insert (struct node * head , int x)
{
    temp = (struct node *) malloc (sizeof (struct node));
    temp -> data = x;
    temp -> link = NULL;
    if (head = NULL)
    {
        head = temp
    }
    else
    {
        temp1 = head;
        while (temp1 -> link != NULL)
        {
            temp1 = temp1 -> link;
        }
        temp1 => link = temp;
    }
    return head;
}
```

```
void main ( )
{
    int p, q, x, i ;
    printf ("Enter the No of elements in linked list - 1 ");
    scanf ("%d", &p);
    for (i=0 ; i<p ; i++)
    {
        printf ("Enter the value ");
        scanf ("%d", &x);
        head1 = insert (head1, x);
    }
    printf("Enter the No of elements in linked list - 2 ");
    scanf ("%d", &q);
    for (i=0 ; i<q ; i++)
    {
        printf (" Enter the value ");
        scanf ("%d", &x);
        head2 = insert (head2 , x);
    }
    temp = (struct node *) malloc (sizeof (struct node));
    temp → link = head 1 ;
    temp → data = head2 → data;
    temp1 = head 1;
    while (temp1 != NULL)
    {
        printf ("%d ", temp→ data);
        temp1 = temp1 → link;
```

```
    }
    printf("\n Linked List 2 \n");
    temp1 = head2;
    while (temp1 != NULL)
    {
        printf ("%d ", temp->data);
        temp1 = temp1->link;
    }
}
```

output:

Enter the No of Elements in Linked List - 1 : 3

Enter the value : 1

Enter the value : 2

Enter the value : 3

Enter the No of Elements in Linked List - 2 :

Enter the value : 4

Enter the value : 5

Enter the value : 6

Linked List 1

4 1 2 3

Linked List 2

5 6