* program to insert and delete an element at nth and kth position in linked list where n and k is taken from user.

code:

```
#include <stdio.h>
# include <stdlib.h>
 struct node
{
 int data;
 struct node *next, *head;
};
void create (int a);
 void insertbeg(int data);
 void insertend(int data);
 void insertmid(int data, int pos);
 void delete(int key);
 void display();
 int main()
{
 int a, k, n, data, pos;
 printf("enter no. of nodes: ");
 scanf("%d", &a);
 create(a);
 printf("The linked list: ");
 display();
 printf("enter the value of n: ");
 scanf("%d", &n);
```

```
if (n==1)
{
printf("\n Enter data to insert at beginning of list : ");
scanf ("%d", &data);
insertbeg(data);
printf ("\n Data in the list \n");
display();
}
else if (n==a)
{
printf ("\n Enter data to insert at end of list : ");
scanf ("%d", &data);
insertend(data);
printf ("\n Data in the list \n");
display();
}
else
{
printf ("\n Enter data to insert at middle: ");
scanf ("%d", &data);
insertmid(data,n);
printf ("\n Data in the list \n");
display();
}
printf ("\n Enter k value: ");
scanf ("%d", &k);
delete(k);
printf ("\n Data in list after deletion \n");
display();
}
return 0;
```

```
void create(int a)
{
  struct node *newNode, *temp;
  int data, i;
  head = malloc(sizeof(struct node));
  printf("Enter data : ");
  scanf("%d", &data);
  head -> data = data;
  head -> next = NULL;
  temp = head;
  for (i=2; i<=a; i++)
  {
    newNode = malloc(sizeof(structNode));
    printf("Enter data: ");
    scanf("%d", &data);
    newNode -> data = data;
    newNode -> next = Null;
    temp -> next = newNode;
    temp = temp -> next;
  }
}
void display()
{
  struct node * temp;
  if (head == NULL)
  {
    printf(" List is empty \n");
    return;
  }
  temp = head;
  while (temp != NULL)
```

```
    {
        printf ("%d ", temp -> data);
        temp= temp -> next;
    }
    printf("\n");
}
void insertbeg(int data)
{
    struct node* newNode;
    newNode = (struct node*) malloc (sizeof(struct node));
    if (newnode ==null)
    {
        printf (" unable to allocate memory ");
    }
    else
    {
        newNode -> data = data;
        newNode -> next = head;
        head = newNode;
    }
}
void insertend (int data)
{
    struct node * newNode, * temp;
    newNode = (struct node *) malloc (sizeof (struct node));
    newNode -> data = data;
    newNode -> next = Null;
    temp = head;
    while (temp -> next != Null)
        temp = temp -> next;
    temp -> next = newNode;
}
```

```
void insertmid( int data, int pos)
{
   int i;
   struct node *newNode, *temp;
     newNode = (struct node*)malloc(sizeof(struct node));
     newNode -> data = data;
     newNode -> next = Null;
      temp = head;
     for (i=2; i<=pos-1; i++)
     {
        temp = temp -> next;
        if (temp == Null)
          break;
     }
     if (temp != Null)
     {
        newNode -> next = temp -> next;
        temp -> next = newNode;
     }
     else
      {
        printf(" unable to insert data in given posit\n");
      }
}
   void delete (int k)
   {
      struct node * prev, *cur;
      while (head != Null && head -> data == k)
      {
        prev = head;
        head = head -> next;
        free (prev);
      return;
      }
```

```
prev = null;
cur = head;
while (cur != Null)
{
    if (cur -> data == k)
    {
        if (prev != Null)
            prev -> next = cur -> next;
        free (cur);
        return;
    }
    prev = cur;
    cur = cur -> next;
}
}
```

output:

```
enter no of nodes: 3
enter data: 1
"    "   : 2
"    "   : 3
the linked list: 1, 2, 3
Enter value of n: 5
Enter data to insert at middle of list: 6
unable to insert data in given position
Data in the list
1 , 2 , 3
Enter the k value: 2
Data in list after deletion is
1, 3.
```

② construct new linked list by merging alternate nodes of two lists.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{ int info;
  struct node * link;
};
struct node* create_list(struct node *);
struct node* concat(struct node *start1, struct node* start2);
struct node* addbeg(struct node* start, int data);
struct node* addend(struct node *start, int data);
void display(struct node * start);
int main()
{
   struct node * start1 = null, * start2 = null;
    start1= create_list(start1);
    start2= create_list(start2);
    printf("first list: ");
    display(start1");
    Print f("second list: ");
    display("strat 2");
    start1 = concat(start1, start2);
    printf("New list: ");
    display(start1);
    return 0;
}
```

```
struct node * concat (struct node *start1, struct node*start2)
{
  struct node * p1;
  struct node * p2;
  struct node * newnode;
  newnode = (struct node *) malloc (size of (struct node));
  if (newnode == null)
  { printf (" unable to allocate memory. ");
  }
  else
  { newnode -> info = p2 -> info;
    newnode -> link = p1;
    p1 = newnode;
  } }
  struct node* create_list (struct node * start)
  { int i, n, data;
    printf (" Enter no. of nodes : ");
    scanf ("%d", &n);
    start = Null;
    if (n==0)
        retun start;
    printf (" enter the data: ");
    scanf ("%d", &data);
    start = addbeg (start, data);
```

```
for (i=2; i<=n; i++)
{
    printf ("enter data: ");
    scanf ("%d", &data);
    start = addend (start, data);
}
return start;
}
void display (struct node * start)
{
    struct node *p;
    if (start == null)
    {
        printf ("list is empty\n");
        return;
    }
    p = start;
    while (p != Null)
    {
        printf ("%d", p -> info);
        p = p -> link;
    }
    printf ("\n");
}
Struct node * addbeg ( struct node * start, int data)
{
    struct node *tmp;
    tmp = (struct node *) malloc (sizeof (struct node));
    tmp -> info = data;
    tmp -> link = start;
    start = tmp;
    return start;
```

```
struct node * addend (struct node *start, int data)
{
    struct node *p, *tmp;
    tmp = (struct node *) malloc (sizeof(struct node));
    tmp -> info = data;
    p = start;
    while (p -> link != NULL)
      p = p -> link;
    p -> link = tmp;
    tmp -> link = null;
    return start;
}
```

output:

```
First list    1  2  3
second list   4  5  6

merged alternate list    1 4 2 5 3 6
```

3) Find all elements in stack whose sum is equa to K (K is given from user)

Code:

```
#include <stdio.h>
int maxsize = 100;
int stack (100))
 int top = -1;
void findpair ( int stack [], int n, int k)
{
   for (int i= 0; i<n-1; i++)
   {
     for (int j= i+1 ; j <n ; j++)
     {
       if (stack [i] +stack [j] ==k)
        {printf (" The pairs are %d and %d \n,"
                                  stack [i], stack (j));
     }   }
}  }
int push (int data)
{
     top = top+1;
     stack[top] = data;
}
int main()
{
  int i, a, x, k;
  printf (" Enter no of elements in stack : ");
  scanf ("%d", &x);
```

```
for (i=0; i<=x-1; i++)
{
    printf ("enter value : ");
    scanf ("%d", &a);
    push (a);
}
printf ("enter k value");
scanf ("%d", &k);
findpair (stack, x, k);
return 0;
}
```

Output:

```
enter no of elements in stack : 4
enter value : 1
enter value : 2
enter value : 3
enter value : 4
enter k value : 5
The pairs are 1 and 4
The pairs are 2 and 3
```

4) write a program to print elements in queue

i) in reverse order

ii) in alternative order.

code:

```
#include<stdio.h>
# define size 20
void enqueue (int);
void display ();
int items [size], front= -1, rear = -1;
int main ()
{
  int x, i, y;
  printf (" Enter no of elements in queue : ");
  scanf ("%d", &x);
  for (i=0 ; i<x; i++) {
  printf (" Enter data: ");
   scanf ("%d", &y);
    enqueue (y);
  }
  display ();
}
void enqueue (int value){
 if (rear == size-1)
    printf ("In Queue is full ");
 else {
     if (front ==-1)
        font = 0;
```

```
    rear ++;
    items[rear] = value;
    printf("\n Inserted -> %d\n", value);
  }
}
void display(){
  int a[20], i;
  if (rear == -1)
    printf("\n Queue is empty !!!");
  else{
    printf("\n Queue elements are:\n");
  for(i=front; i<=rear; i++){
    a[i]=items[i];
    printf("%d\t", items[i]);
  }
}
printf("\n Queue elements in reverse:\n");
for(i=rear; i>=front; i--)
{ printf("%d\t", a[i]);
}
printf("\n queue elements in alternate order:\n");
for(i=front; i<=rear; i+=2)
{ printf("%d\t", a[i]);
}
}
```

output :

```
Enter no of elements in queue : 4
Enter data: 1
inserted -> 1
Enter data: 2
inserted -> 2
Enter data: 3
inserted -> 3
Enter data: 4
inserted -> 4
Queue elements are:
1   2   3   4
Queue elements in reverse order :
4   3   2   1
Queue elements in alternate order :
1   3
```

(5) (1) How array is different from linked list.

Differences:-

a) Arrays have fixed size where as linked list not.

b) we cannot access previous element directly through linked list.

c) There is no binary search for singly linked list whereas arrays does.

d) elements of arrays can be modified easily by identifying index value whereas it is complex for linked list.

e) Array elements can't be added or deleted once it is declared but for linked list it can be added and deleted.

f) Array supports Random access, accessing elements in an array is fast with constant time complexity of O(1) whereas sequential, accessing nth element of linked list, time complexity is O(n).

(ii) write program to add first element of one list to another list.

code:

```
# include<stdio.h>
# include<stdlib.h>
  struct node
  {
    int info;
    struct node *link;
  };
  struct node * create_list(struct node *);
  Struct node * list(struct node * start1, structve node * start2);
  struct node * addbeg(struct node *start, int data);
  struct node * add end(struct node * start, int data);
  void display(struct node * start);
  int main()
  {
    struct node * start1 = null, * start2 = null;
    start1 = create_list(start1);
    start2 = create_list(start2);
    printf("First list : ");
    display(strat1);
```

```
printf ("second list : ");
display (start2);
start1 = list (start1, start2);
printf ("New list : ");
display (start1);
return 0;
}

struct node * list (struct node * start1, struct node * start2);
{
struct node * p1;
struct node * p2;
struct node * newNode;
p1 = start1;
p2 = start2;
newNode = (struct node *) malloc (sizeof(struct node));
if (newnode == Null)
{
printf ("unable to allocate memory ");
}
else
{ newnode -> info = p2 -> info;
newnode -> link = p1;
p1 = newNode;
} }
```

```
struct node *create_list (struct node *start)
{
  int i,n,data;
  printf ("Enter no of nodes: ");
  scanf ("%d", &n);
  start = Null;
  if (n==0)
   return start;
  printf ("Enter data: ")
  scanf ("%d", &data);
  start=addbeg (start,data);
  for (i=2; i<=n; i++)
  {
    printf ("Enter the data: ");
    scanf ("%d", &data);
    start= addend (start,data);
  }
   return start;
}
  void display (struct node * start)
  {
     struct node *p;
     if (start ==null)
     {
       printf ("list is empty \n");
       return;
     }
```

```
p= start;
while (p!= Null)
{
  printf("%d", p->info);
  p= p-> link;
 }
 printf ("\n");
}

struct node * addbeg (struct node * start, int data)
{
 struct node * tmp;
 tmp= ( struct node *) malloc(sizeof(struct node));
 tmp -> info=data;
 tmp -> link = start;
 start=tmp;
  return start;
}
struct node *addend(struct node *start, int data)
{
 struct node * p ,* tmp;
 tmp = ( struct node *) malloc (sizeof(struct node));
 tmp -> info =data;
 p = start;
 while (p -> link != Null)
    p = p -> link;
  p-> link= tmp;
  tmp-> link = Null; return start;
}
```

output:

```
Enter the no of nodes : 4
Enter the data : 1
   "   "   "  4
   "   "   "  6
   "   "   "  7
Enter no of nodes : 2
Enter the data : 2
   "   "   " : 3
First list : 1 4 6 7
Second list : 2 3
New list : 2 1 4 6 7
```