1)

/* Program to insert and delete an element at the nth and kth position in a linked list where n and k are taken from user. */

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

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

struct node* head ;
void insert (int data, int data)
{
Node* temp = new node () ;
temp -> next = Null ;
if (n == 1)
{
 temp -> next = head ;
 head = temp;
 return ;
 }
void delete
{
Stut Node* temp = head ;
if (k == 1)
 {
head = temp -> next ;
return ;
 }
Node* temp = head ;
```

```
for (int j=0 ; j<n-2 ; j++)
  temp = temp -> next;
 }
 temp->next = temp->next;
 temp -> next = temp;
}
void print();
for (int j=0 ; j<k-2 ; j++)
 temp = temp -> next;
 free(temp);
 }
int main()
{
int n, e, k;
head = NULL;
Printf ("Give the position for inserting");
scanf ("%d", &n);
scanf ("%d", &e);
insert (e, n);
Printf ("Give the position for deletion");
scanf ("%d", &k);
delete (k);
Printf (e);
return;
}
```

2)

/* c program to construct a new linked list by merging alternative nodes and two lists. */

Sol°-

```
#include <stdio.h>
#include <stdlib.h>
struct node {
 int data;
 struct node *next;
};
Void Print list (struct node *head)
{
 Printf ("%d →", ptr→data));
  ptr = ptr→next;
  Print f ("Null \n");
 }
Void Push (Struct node *head, int data)
{
Struct node * new = (Struct node)
malloc (sizeof (struct node));
new→ data = data;
new→ next = * head;
 * head = new;
}
```

```
struct node *merge (struct node *l, struct node *m)
{
struct node false; false
 struct node *fail = &false;

 false.next = NULL;
 while (1)
 {
 if (l == NULL)
 {
 fail->next = m;
 break;
 }
 else if (b == NULL)
 {
 fail->next = l;
 break;
 }
 else
 {
 fail->next = l;
 fail = l;
 l = l->next
 fail->next = m;
 }
 }
 return fail.next;
}
```

```
void main()
{
int keys[] = {1,2,3,4,5,6,7};
int n = sizeof(keys)/sizeof(keys[0]);
struct node* l = NULL; *m = NULL;
for(int j = n-1; j>0; j = j-2)
    push(&l, keys[j]);
for(int j = n-2; j>=0; j = j-2)
    push(&m, keys[j]);
struct node*head = merge(l,m);
printlist(head);
}
```

3.

```
/*C program to find all the elements in
the stack whose sum is equal to k*/
#include<stdio.h>
#include<stdio.h>
int top=-1;
int a;
char stack[100];
void push(int a);
char pop();
int main()
{
int i,n,x,y,z,k,sum=0,count=1;
printf("Give the no.of elements");
```

```
scanf("%d", &n);
for(j=0; j<n; j++){
printf("Give the next element ");
scanf("%d", &x);
push(x);
}
printf("Provide the sum to verify");
scanf("%d", &k);
for(j=0; j<n; j++)
{
y= pop();
sum = sum+y;
count = count+1;
if(sum==k){
for(int i=0; i<count; i++)
printf("%d", stack[i]);
a=1;
break;
}
push(y);
}
if(a!=1)
printf("It is not equal to the sum");
}
void push(int a)
{
if(top==99)
{
printf("\n stack is full");
return;
}
```

```
top+= 1;
stack [top]= a;
}
char pop()
{
if (stack[top]==-1)
{
printf("The stack is fully empty");
return 0;
}
a = stack[top];
top= top-1
return a;
}
```

---

4. /*C program to print the elements in
queue i. in reverse order
ii. in alternate order.

```
#include<stdio.h>
#define size 20
void insert(int);
Void delete();
int queue[20], x=-1, y=-1;
Void main()
{
int value, choice;
while(1)
{
Printf("1.Insertion");
Printf("2. Deletion");
printf("3. Print reverse");
Printf("4. Print Alternative");
Printf("5. Exit");
Printf("chose one option");
Scanf("%d", &choice);
So
```

```
switch (choice){
 case 1;
 Printf ("Value to Insert ");
  scanf ("%d", &value);
  insert (value);
  break;
 case2;
Printf ("Give the deletion value)
 delete ();
  break;
 case3;
 Printf ("The reversed queis ");
 for (int j=size; j>=0; j--)
{
  if (queue[j]==0)
   continue;
  Printf ("%d", queue[j]);
 }
  break;
  case4,
  Printf ("Alternate elemet is");
   for (int j=0; j<size; j+=2)
   {
   if (queue[j]==0)
    continue;
    Printf ("%d", queue[i]);
  }
   break;
  case5 :
  exit(0);
  default
  Printf ("wrong option");
  }
```

```
void insert(int value){
 if ((x==0&&y==size-1)||x==y+1)
 Printf("\n The insertion is not at all posible");
  else{
  if (x==-1)
   x=0;
     y=(y+1)%size;
     queue[y]=value;
     Printf("\n Insertion is done");
  }
  }
  Void delete(){
   if (x==-1)
   Printf("deletion is not at all posible");
   else{
    Printf("%d", queue[x]);
     x=(x+1)%size;
     if(x==y)
     x=y=-1;
     }
     }
```

5)

i. How array is different from the linked list.

The Major differencese between array and linked list are:-

i) Arrays are index based data structure where each element is assosiated with an index. And the linked list relies on references where each node consists of the data and the reference to the

previous and next element.

99) Elements are sorted consecutively in array whereas it is sorted randomly in linked list.

99) /* c Program to add the first element of one list to another list */

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
int data;
struct node* next;
};
void push(struct node **head_ref,
          int dewy_data)
{
struct node* dewy_data = (struct node*)
 malloc(sizeof(struct node));
dewy_node -> data = dewy_data;
dewy-node -> next = (*head_ref);
(*head_ref) = dewy_node;
}
void printlist(struct node* head)
{
while(temp = NULL)
{
printf("%d", temp -> data);
temp = temp -> next;
}
```

```
void merge(struct node* x, struct node** y);
{
struct node* x_curr = x, *y_curr = *y;
struct node* x_next, *y_next;
while (x_curr != NULL && y_curr != NULL)
{
    x_next = x_curr->next;
    y_next = y_curr->next;
    y_curr->next = x_next;
    x_curr->next = y_curr;
    x_curr = x_next;
    y_curr = y_next;
}
*y = y_curr;
}
int main()
{
struct node* x = NULL, *y = NULL;
push(&x, 1);
push(&x, 2);
push(&x, 3);
printf("The only first linked list is:");
print list(x);
push(&q, 4);
push(&q, 5);
push(&q, 6);
Print("second linked list");
print list(y);
```

```
merge(x, &y);
Printf("corrected first linked list");
Print list = (p);
Printf("corrected second linked list");
Print list = (q);
getchar();
return(0);
}
```

Thank You