Programs

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 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 n) {
Node* temp = new node();
temp->data = data;
temp->next = NULL;
if (n==1) {
temp->next=head;
head=temp;
return;
}
Void delete (int p);
Struct node* temp = head;
if (p==1) {
head = temp->next;
```

```
free(temp);
return;
}
Node* temp = head;
for(int i=0; i<n-2; i++) {
temp= temp->next;
}
temp->next = temp->next;
temp->next = temp;
}
void print();
for(int i=0; i<p-2; i++) {
temp = temp->next;
free(temp);
}
int main() {
int n, c, p;
head = NULL;
printf("Enter the position of insertion);
scanf("%d", &n);
scanf("%d", &c);
insert(c, n);
printf("Enter the position of deletion);
scanf("%d", &p);
Delete(p);
```

```
Print (c,0);
return;
}
```

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

```
#include <stdio.h>
#include<stdlib.h>
Struct node
{
    int data;
    Struct data node* next;
};
Void printlist (struct node*head)
{
    Struct node *ptr = head;
    while (ptr)
    {
    printf ("%d→ ", ptr→data);
     ptr = ptr→next;
    }
    printf ("NULL\n");
}
```

```
Void Push (struct node** head, int data)
{
  Struct node* newnode = (struct node*) malloc
                         (size of(struct node));
  newnode→data = data;
  newnode→next = *head;
  *head = newnode;
}
Struct node *shuffle merge (struct node* a, struct node* b)
{
  Struct node dummy;
  Struct node* tail = &dummy;
  dummy.next = NULL;
  while (1)
  {
    if (a==NULL) {
      tail→next = b;
      break;
    }
    else if (b==NULL){
      tail→next = a;
      break;
    }
    else {
      tail→next = a;
```

```
        tail=a;
        a=a->next;

        tail->next=b;
        tail=b;
        b=b->next;
    }
    return dummy.next;
}
int main(void)
{
  int keys[]={1,2,3,4,5,6,7};
  int n= size of keys / size of(keys[0]);
  Struct node *a=NULL, *b=NULL;
  for(int i=n-1; i>=0; i=i-2)
     push(&a, keys[i]);
  for(int i=n-2; i>=0; i=i-2)
     push(&b, keys[i]);

  printf("First list");
  printlist(a);

  printf("Second list");
  printlist(b);
```

```
struct node *head = shufflemerge(a,b);
printf("After merge");
printlist(head);
return 0;
}
```

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

```
#include <stdio.h>
int top=i-1;
int x;
char stack[100];
void push(int x);
char pop();
int main()
{
  int j,i,n,b,d,k,g,sum=0,count=1;
  printf("Enter the number of elements in Stack");
  scanf("%d",&n);
  for(i=0;i<n;i++){
  printf("Enter next element");
  scanf("%d",&b);
  push(b);
}
```

```
printf("Enter the sum for check");
scanf("%d", &k);
for(i=0; i<n; i++) {
d = pop();
Sumd=d;
Countd=1;
if (sum==k) {
for(int j=0; j<count; j++)
printf("%d", stack[i]);
g=1;
break;
}
push(d);
}
if (g!=1)
printf("Elements in stack will nod add to the sum");
}
Void push (int x)
{
  if (top==99)
  {
  printf("\n Stack is full \n");
  return;
} }
top = top+1;
```

```
Stack [top]=x;
}
char pop() {
if (stack[top]==-1) {
printf("\n Stack is empty \n");
return 0;
}
}
x= stack[top];
top=top-1;
   return x;
}
```

4. Write a program to print the elements in a queue:
(i) In reverse order
(ii) In alternate order

```
#include <stdio.h>
#define size 10
void insert(int);
void delete();
int queue[10], f=-1, r=-1;
void main()
{
```

```
int value, choice;
while(1){
    printf("\n\n **** MENU **** \n");
    printf("1.Insertion\n 2.Deletion\n 3.Print reversed
        4.Print Alternate\n 5. exit");
    printf("\n Enter your choice");
    scanf("%d", &choice);
    switch(choice){
        case 1:
        printf("Enter the value to insert");
        scanf("%d", &value);
        insert(value);
        break;
        case 2:
        delete();
        break;
        case 3:
        printf("The Reversed queue is");
        for(int i=size; i>=0; i--)
    {
        if(queue[i]==0)
        continue;
        printf("%d", queue[i]);
}
        break;
```

```
case 4:
    printf("Alternate elements of queue are");
    for(int i=0; i<size; i+=2)
{
    if (queue[i]==0)
    continue;
    printf("%d", queue[i]);
}
    break;
    case 5:
     exit(0);
    default:
     printf ("\n Wrong selection");
    }
}}
Void insert (int value){
  if ((f==0 && r==size-1) || f==r+1)
   printf("\n Queue is full and insertion can't be done");
  else {
    if(f==-1)
f=0;
    r=(r+1)%size;
    queue[r]=value;
    printf("\n Insertion success");
}}
```

```
Void delete() {
  if (f==-1)
     printf("\n Queue is Empty and Deletion can be done");
  else {
     printf("\n Deleted %d", queue[f]);
     f=(f+1)%size;
     if (f==n)
f=n=-1;
}}
```

5. (i) How array is different from linked list ?

The major difference between array and linked list regards to their structure. Arrays are index based data structure where each element associated with an index. But in linked list, relies on refrences where each node consists of the data and the refrences to the previous and next element.

(ii) Write a 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 printlist (struct node *head)
{
    struct node *ptr=head;
    while(ptr)
    {
     printf("%d→", ptr→data);
     ptr = ptr→ next;
}
    printf("NULL\n");
}
Void Push (struct node** head, int data)
{
    struct node *newnode = (struct node*) malloc
                    (size of (struct node));
    new node→ data = data;
    new node→ next=*head;
    *head = new node;
}
Void move node (struct node **dest ref, struct node**
                                    Source ref)
{
```

```
    if (*Sourceref == NULL)
      return;
    struct node* newnode = *sourceref;
    *sourceref = (*sourceref)→next;
    newnode→next = *destref;
    *destref = newnode;
}
int main(void)
{
  int keys[] = {1,2,3};
  int n = size of(keys) / size of(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]);

  Move node(&a, &b);

  printf("first list");
  printlist(a);
  printf("second list");
  printlist(b);
  return 0;
}
```