CSE-G. DSA ASSIGNMENT-4 D. Harika AP19110010499.

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

```
#include<stdio.h>
Struct node
{
  int data;
  Struct node *next;
} *head;
  void create(int a);
  void insert beg (int data);
  void insert end (int data);
  void insert mid (int data, int pos);
  void delete (int key);
  void display ();
  int main ()
  {
  int a, k, n, data, pos;
  Printf ("Enter number 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 List:");
  Scanf ("%d", &data);
  insert beg (data);
  printf ("\n Data in the list \n");
  display ();
  }
  else if (n==a)
  {
```

```
printf("\n enter data to insert at end of the list : ");
scanf("%d", &data);
insert end (data);
printf("\n data in the list \n");
display();
}
else
{
printf("\n enter data to insert at middle of the list");
scanf("%d", &data);
insert mid (data, n);
printf("\n data in the list \n");
display();
}
printf("\n enter the 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 (size of (struct node));
  printf("enter data : ");
  scanf("%d", &data);
  newnode -> data = data;
  newnode -> 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 insert beg (int data)
{
Struct node *newnode;
new node = (Struct node*) malloc (size of (struct node));
if (new node ==null)
{
 printf ("unable of allocate memory ");
}
else
 {
  new node -> data =data;
  newnode->next =head;
  head = new node;
 }
}
void insertend (int data)
{
 Struct node * new node , *temp;
  new node = (struct node*) malloc (size of (struct node));
  new node -> data=data;
  new node -> next =null;
  temp =head;
 while (temp -> next! =NULL)
      temp= temp->next;
       temp-> next =newnode;
}
void insert mid (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 position\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;
}
}
```

Out put

```
Enter number of nodes: 3
enter data : 2
enter data : 4
Enter data : 55
The linked list: 2,4,55
Enter the value of n: 4
enter data to insert at middle of the list: 5
Data in list: 2,4,55,5.
Enter the k value: 4
Data in List after deletion: 2,55,5.
```

2. Construct a new linked list by merging alternate of two lists for example in list1 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>
struct node
{
 int info;
 struct node* link;
 };
struct node *create_list (struct node*);
struct node* concat (struct node *start1, struct node *start2);
Struct node * add beg (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);
printf ("second list :");
display (start2);
start1 = concat(start1, start2);
printf ("concatenated list : ");
display(start1);
return 0;
}
struct node* concat(struct node * start1, struct node *start2)
{
  struct node *ptr;
  if (start1 == NULL)
  {
   start1 == NULL start2;
   Return start1;
  }
  if (start2 == NULL)
     return start1;
   ptr = start1;
  while (ptr -> link != NULL)
    ptr = ptr -> link;
 ptr -> link = start2;
  return start1;
}
struct node * create_list (struct node *start)
{
 int i, n, data;
 printf ("enter the no. of nodes : ");
 scanf ("%d", &n);
 start = NULL;
 if (n == 0)
   return start;
   printf ("enter the data:");
   scanf ("%d", &data);
    start = add beg (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 *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: 2
Enter the data: 6
Enter the data: 8.

3. find all the elements in the stack whose sum is equal to k

```
#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;
}
```

4. write a program to print the elements in a queue.
1. in reverse order
2. in alternate order

```
#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("\n queue is full!!");
 else{
    if(front==-1)
     front=0;
    rear++;
    items(rear)=value;
    printf("\nInserted -> %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+=2)
{
printf ("%d\t", a[i]);
}
}
```

5. i) How array is different from the linked list. difference between array and linked list.

1) An array is a data structure type data elements where as the linked list is considered as non-primitive data structures contains a collection of unordered linked elements known as nodes.

2) In the array the elements belong to indexes, i.e; if you want to get into the fourth element you have to write the variable name with its index or location within the square bracket.

3) In a linked list through, you have to start from the head and work your way through until you get to the fourth element.

4) According add element in an array is fast while in linked list takes linear time, so it is quit a bit slower.

5) Operations like insertion and deletion in array consume a lot of the time. on the other hand the performence of these operations in linked list is fast.

ii) write a program to add the first element of one list to another list for example we have {1,2,3} in list 1 and {4,5,6} in list 2 we have to get {4,1,2,3} as output for list 1 and {5,6} for list 2.

```
#include <stdio.h>
struct node
{ int info;
  struct node* link;
```

```
};
Struct node * create_list (struct node*);
Struct node * list (struct node * start 1, Struct node *start 2);
Struct node* add beg (struct node * start, int data);
Struct node * addend (struct node * start, int data);
void display (struct node *start);
int main()
{
  Struct node *start1= NULL *start 2= NULL;
  Start 1 = create_list (start 1);
  Start 2 = create_list (start 2);
  Printf ("first list :");
  display (start 1);
  Start1 = list (start 1, start 2);
  Printf ("new list :");
  display (start 1);
  return 0;
}
Struct node * list (Struct node* start1, Struct node * start 2)
{
 struct node *P1;
 Struct node *P2;
 Struct node *New node;
 P1 =Start 1;
 P2 = Start 2;
 new node =(struct node*) malloc (sizeof (struct node));
 if (new node == null)
{
 printf ("unable to allocate memory");
}
else
 {
 new node ->info =P2->info;
 new node -> link = P1;
  P1 = new node;
 }
}
Struct node * create_list (struct node *start)
```

```
{
int i,n,data;
printf("enter the no.of nodes:");
scanf("%d",&n);
start=NULL;
if(n==0)
  return start;
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;
}.
```