AP19110010532 (Assignment-4) Pragya Gupta CSE-A

Q1 Write a program to insert and delete an element at the nth and Kth position in a linked list where n and K is taken from user.

```
#include <stdio.h>
# include <stdlib.h>
void pos(node*, int, int)
int size = 0;
struct node { int data;
struct node * next;
}
node *getnode(int data)
{
  node * newnode = (struct node *) malloc(newnode);
  new_node => data = data;
  new_node → next = null;
  return new-node;
}
void insert(node *current, int position, int data)
{ if(position < 1 || position > size + 1)
    printf("Wrong input");
  else
  { while(position--)
    { if(position == 0)
  {
    node * temp = getnode(data);
    temp → next = *current;
    *current = temp;
    }
  else { current = &(*current) → next;
  }
```

```
size++;
}
}
void printf(struct node *head)
{ while(head != NULL)
{ printf("%d", head->data);
*head = head->next;
}
printf("\n");
}
Void delete(struct node *head, int pos)
{ if(head == NULL)
return;
temp = head;
if(pos == 0)
{
*head = temp->next;
free(temp);
return;
for(int i=0; temp != NULL && i<pos-1; i++)
temp = temp->next;
free(temp->next);
temp->next = next;
} int main()
{ struct node *head = NULL;
push(&head, 7);
push(&head, 8);
push(&head, 6);
insert(&head, 7, 15);
delete(&head, 4);
printlist(head);
return 0;
}
```

2. Construct a new linked list by merging alternate nodes of two lists for example in list1 we have {1,2,3} and in list2, we have {4,5,6} in the new we should have {1,2,3,4,5,6}

```
#include <stdio.h>
#include <stdlib.h>
struct node {
int data;
struct node *next;
}
void print_list(struct node *head)
{
 struct node *ptr = head;
 while (ptr)
 {
  printf("%d", ptr -> data);
  ptr = ptr -> next;
 }
 printf("NULL");
 }
 void push (struct node *head, int data)
 {
 struct node *new = (struct node *) malloc (size of (struct node));
new -> data = data;
new -> next = *head;
*head = new)
}struct node *merge (struct node *a, struct node *b)
{ struct node ml;
  struct node *tail = ml;
  ml.next = NULL;
 while (1) {
```

```
if(a==NULL)
{
  fail -> next=b;
  break;
}else
  {
    fail->next=a;
     fail=a;
     a=a->next;
     fail->next=b;
}}return ml.next;
}
void main()
{
  int key[]={1,2,3,4,5,6,9};
  int n = size of(keys)/size of(key[0];
  struct node *a=NULL; *b=NULL;
for(int i=n-1; i>0;i=i-2)
    push(&a,key[i]);
for(int i=n-2; i>=0; i-2)
    push(&b,key[j]);
struct node * head=merge(a,b);
print_list(head);
    }
```

4. Write a program to print all the elements in a queue.

(i) reverse order (ii) alternate order

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
  int data;
  struct node *next;
};
void print_rev(struct node *head)
{
  if (head == NULL)
    return;
  print_rev(head->next);
  printf("%d ", head->data);
}
void push(struct node *head, char n)
{
  struct node *node_n = (struct node*) malloc(sizeof(struct node));
  node_n->data = new;
  node_n->next = head;
  *head = node_n;
}
int main()
{
  struct node *head = NULL;
  push(&head, 4);
  push(&head, 3);
  push(&head, 2);
  print_new(head);
  print alternate(head);
  return 0;
}
```

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

```
#include <stdio.h>
void find (int a[], int n, int p)
{ int sum = 0;
  int l = 0, h = 0;
  for( l=0; l<n; l++)
    { while(sum<p && h<n)
      sum = sum + a[h];
      h++;
  if (sum == p)
  { printf("Found");
    return
    sum -= a[l];
  }}
void main ()
{ int a[ ] = {2, 6, 0, 9, 7, 3}
  int p = 15;
  int n = sizeof(a)/ sizeof(a[0]);
  find (a, n, p);
  return 0;
}
```

```
void printalternate(struct node *head)
{
int count=0;
while(head != NULL)
{ if(count % 2 == 0)
    count << head -> data << " ";
  count++;
  head = head -> next;
}
```

5 i How is array diff from linked list.

Array	Linked list
a) An array is a data structure that contains a collection of similar type data elements.	a) A linked list is a non-primitive data structure contains a collection of unordered linked elements known as nodes.
b) In array elements belong to respective index so easy to search.	b) In linked list you have to traverse starting from head till the element given.
c) Here you can access to the element very fast.	c) Here access to the element is slow.
d) Operation like insertion & deletion in array is easy.	d) Operation like insertion & deletion takes time.

ii)
```
#include <stdio.h>
#include <stdlib.h>
int len(int a[])
{ int i=0, an=0;
  while(true);
  {
    if(a[i])
      {an++, i++;}
  }
  else
  { break;
  }
  }return an;
}
void changinglist(int a[], int b[])
  { for(int i = len[a]-1; i>=0; i--)
  { a[i+1] = a[i];
  }
a[0] = b[0];
printf("Elements are:");
for(int i=0; i< len(a); i++)
  { printf("%d", a[i]);
  }
for(int i=0; i<len(b); i++)
  { b[i] = b[i+1];
  } printf("Elements are:");
for(int i=0; i<len(b); i++)
{ printf("%d", b[i]);
}
}
```

```
int main()
{ int a[10] = {1,2,3};
  int b[10] = {4,5,6};
  changinglist(a,b);
}
```