DSA - ASSIGNMENT (4)

M. R. M. Sitarama Reddy
AP19110010363
CSE-F

(1) Write a program to insert & delete an element at the n^{th} & k^{th} position in a linked list where n & 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;
   }
   Node *tempo = head;
   for(i=0; i<n-2; i++)
   {
     tempo = tempo->next;
    }
    temp -> next = tempo -> next;
    tempo -> next = temp;
}
```

```
void delete (int k)
{
   Struct Node * temp = head;
   if (k==1)
   {
     head = temp -> next;
     free(temp);
     return;
    }
    Node * temp = head;
    for (i=0; i<n-2; i++) {
     temp = temp -> next;
     }
    temp -> next = temp -> next;
    temp -> next = temp;
}
```

```
int main( )
{
  int n, x, k;
  head = NULL;
  Printf ("Enter position and element for insertion: ");
  Scanf (" %d", &n);
  Scanf (" %d", &x);
  insert (x, n);
  printf (" Position of deletion: ");
  Scanf (" %d", &k);
  delete(k);
  print(x)
  return;
}
```

(2) consider the new linked list by merging alternate node of two list for example in list 1 {1,2,3} & list 2 {4,5,6} in new list {1,4,2,5,3,6}.

```
#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;
   printf ("NULL/n");
 }
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 temp;
  Struct Node *tail = &temp;
  temp . 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;
        }
    }return fake->next;
}
void main()
{
    int keys[] = {1,2,3,4,5,6,7}
    int n = sizeof(keys) / sizeof key[a]
    Struct Node *a = NULL; *b = NULL;
    for(i=n-1; i>0; i=i-2)
    push(&a, key[i]);
    for(i=n-2; i>=0; i=i-2),
    push(&b, key[i]);
    Struct Node *head = merge(a,b);
    printlist(head);
}
```

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

```
#include <stdio.h>
int top=-1;
int x;
char stack[100];
void push(int x);
char pop();
int main()
{
int i,n,a,t,k,f,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",&a);
push(a);
}
printf("Enter the sum to be checked");
scanf("%d",&t);
for(i=0;i<n;i++){
t=pop();
sum+=t;
count+=1;
if(sum==k){
for(int j=0;j<count;j++)
printf("%d",stack[j]);
f=1;
break;
}
push(t);
}
if(f!=1)
printf("Elements in stack dont add up to 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 Empty Stack\n");
       return 0;
   }
  x = stack[top];
 top = top-1;
 return x;
}
```

(4) Write a program to print the elements in a queue

```
#include <stdio.h>                    // reverse order
# define SIZE 10                      // alternate order
void insert(int);
void delete();
int queue[10], f = -1, r = -1;
void main()
{
int value, choice;
while(1)
{
printf("\n menu\n");
printf(" 1. Insertion \n 2. Deletion \n 3. Reverse \n 4. Alternate ");
Printf (" \n Enter choice: ");
Scanf (" %d", &choice);
Switch(choice)
{
case 1: printf("Enter value to insert: "),
        Scanf ("%d", &value);
        insert (value);
        break;
case 2: delete();
        break;
case 3: printf(" Reverse queue:");
        for (i=SIZE; i>=0; i--)
        {
        if(queue[i] ==0
          continue
          print ("%d", queue[i]);
        }
        break;
Case 4: printf ("Alternate elements of queue: ");
        for (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! Try again! ");
        }
    }
}

void insert (int value){
 if ((f == 0 && r == SIZE - 1) || f == r+1)
 printf ("\n Queue is full");
 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");
 else
 {
   printf ("\n Delete : %d", queue[f]);
   f = (f+1) % SIZE;
   if f = r = -1;
 }}
```

(5)

i. How array is different from the linked list?

The major difference b/w Array and Linked list regard to their structure. Arrays are index based data structu where each element associated with an index. On the other hand, linked list relies on reference to th previous and next element.

ii.

```
#include <stdio.h>
#include <stdlib.h>
Struct Node
{
 int data;
  Struct node * next;
};
Void push (struct Node ** head ref, int new_data)
{
Struct node * new_node =(Structnode *) malloc (size of
                                                Struct no
 new_node → data = new_data;
 new_node → next = (*head_ref);
 * head-ref_ = new-node;
 }
 Void printlist (Struct node * head)
 {
   Struct node *temp = head;
   while (temp! = NULL)
   { printf (" %. d", temp → data),
     temp = temp → next;
   }
   print (" \n"),
}
```