Assignment-4

Sunkavalli Teja
AP19110010483
CSE-F

① Write a program to insert and delete an element at n^{th} and k^{th} Position in a linked list where n, K is taken from user.

Pro

```
#include <stdio.h>
#include <stdlib.h>
Void ans (node*, int, int)
int size=0;
struct node {
    int data;
    struct node*next;
}
node*get node (int data)
{
    node*newnode = (struct node*)malloc
                    (newnode);
    newnode → data = data;
    newnode → next = null;
    return new node;
}
Void ins (node*current , int pos , int data)
{
    if (pos<1 || pos > size+1)
        Printf ("Invalid");
    else {
        while (pos--)
```

```
        {
            if (pos==0)
            {
                node *temp=get node(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 del (struct node* head-del , int pos)
{
   if (head-del == NULL)
       return;
       temp= head_del;
     if (pos=0)
     {
        *head_del = 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, 10);
    push(&head, 8);
    push(&head, 12);
    ins(&head, 7, 9);
    del(&head, 8);
    printlist(head);
    return(0);
}
```

Output: 10, 12, 7, 9.

(2) Construct a new ~~node~~ linked list by merging alternate nodes of two list for example in list 1 we have {1,2,3} and in list 2 we have {4,5,6} The new linked list should be {1,4,2,5,3,6}

Prog

```
#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 *)
                    mallo (sizeof (struct node));
    newnode -> data = data;
    newnode -> next = next;
    *head = newnode;
}

Struct node * shuffleMerge (struct node* a,
                    struct node *b)
{
```

```
        structnode dummy;
         Struct node * tail = &dumy;
         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 = sizeq(keys)/ size q (keys[0]);

     Structnode *a = NULL , *b=NULL;
      for(inti =n-1 ; i >=0; i =i-2)
```

```
        Push(&a, keys[i]);
    for (int i=n-2; i >=0; i=i-2)
        Push(&b, key[i]);

    Printf("First List:");
     Printf List(a);
    printf("second List:");
     Printf List(b);
    Structnode*head = shuffleMerge(a,b)
     Printf("After merge:");
     Printf(head);
      return 0;
}
```

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

```
#include <stdio.h>
Void find (int arr[], int n, int s){
        int sum=0;
        int l=0, h=0;
        for (l=0; l<n; l++){
                while (sum<s && h<n)
                        sum = sum + arr[h];
                        h++;
```

```
        if (sum==s)
        {
            printf ("found");
            return;
        }
        sum = sum - arr[i];
    }
}
int main (Void)
{
    int arr[] = {2, 0, 1, 5, 18, 20};
    int s = 10;
    int n = sizeof(arr) / sizeof(arr[0]);
    find (arr, n, s);
    return 0;
}.
```

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

(i) In reverse order,

(ii) In alternate order.

Prog

```
#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);
    Print ("%d", head → data);
Void push (struct node * head rev, char new)
{
  struct node *node_new = (struct node *)
                          malloc (sizeof (struct node));

  node_new → data = new;
  node_new → next = (head*_ref);
  (*head_ref) = node_new;
}
int main()
{
    struct node *head = NULL;
    push (& head ;4);
    Push (& Head, 21);
     Push (&head, 13);
    print new (head); print alternate (head);
    return 0;
}
Void print alternate (struct node * head)
{
```

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

(5) (a) Difference between Array, linked list:

(1) An array is collection of similar data types., where as linked list is a non-primitive data structure contains a collection of unordered linked elements known as nodes.

(2) In array memory is assigned during compile time while in linked list it is allocated during execution of runtime.

5. (6)
```
#include <stdio.h>
#include <stdlib.h>
int len(int a[]);
{
    int i=0;  a,n=0;
```

```
        while (1)
        {
            if (a[i])
            {
                an++, i++;
            }
            else
            { break;
            } }
            return an;
        }
Void changing list (int a[], int b[])
{
    for (int i = len(a)-1; i>=0; i--)
    {
        a[i+1] = a[i];
    }
    a[0] = b[0];
    Printf ("\n the elements of first array
                                    :\n");
    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 (" element in second array \n");
    Printf ("%d", b[i]);
}
int main(){
    int a[10]={1,2,3}, b[10] = {4,5,6};
    changing list (a,b);
}.
```

Lab Programs

① C code for Depth-First search using arrays.

Prog

```
#include <stdio.h>
int G[10][10], visited[10], n;
Void DFS(int i)
{
    int j;
    Printf("\n %d", i);
    Visited[i] = 1;
    for(j=0; j<n; j++)
        if(!visited[i] && G[i][j]==1)
            DFS(j);
}
Void main()
{
    int i, j;
    Printf("Enter number of vertices");
    Scanf("%d", &n);
    Printf("\n enter matrix for graph:");
    for(i=0; i<n; i++)
    {
        for(j=0; j<n; j++)
        {
            scanf("%d", &G[i][j]);
        }
    }
    for(i=0; i<n; i++)
        Visited[i]=0;
    DFS(0);
}
```

(2) C program for Breadth_First-search (BFS) using arrays.

Prog:

```
#include <stdio.h>
int a[20][20], q[20], visited[20], n, i, j, f=0, r=-1;

Void BFS(int v)
{
        for(i=1; i<=n; i++)
                if (a[v][i] && !visited[i])
                        q[++r] = i;
                if (f<=r)
                {
                        visited[q[f]] = 1;
                        BFS(q[f++]);
                }
}

Void main()
{
    int v;
    Printf("\n enter number of vertices");
     scanf("%d", &n);
     for(i=1; i<=n; i++)
      {
            q[i]=0;
           visited[i]=0;
      }
     Printf("\n enter graph data in matrix");
      for(i=1; i<=n; i++)
```

```
        for(j=1; j<=n; j++)
                scanf("%d", &a[i][j]);

    printf("\n enter starting Vertices");
    scanf("%d", &v);
     BFS(v);
     Printf("\n The node which
                are reachable are:\n");
     for(i=1; i<=n; i++)
            if (visited[i])
                printf("%d\t", i);
             else
                printf("BFS not possible");
}.
```