Assignment - 4

Sunkavalli Tulasi
AP19110010452
CSE-F .

1) C program to insert

```
#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    strut node *neaet;
}*head;
void create_list(int p);
void insert (int n, int data);
void delete (int k);
void traverse();
int main(){
    int p,n,data,k;
    printf("Enter the number of nodes");
    scanf("%d", &p);
    printf("Enter the position to insert");
    scanf("%d",&n);
    printf("Enter the data of inserted node");
    scanf("%d", &data);
    printf("Enter the position of node to delete");
    scanf("%d", &k);
    create_list(p);
    insert(n,data);
    delete(k);
    traverse();
```

```
    return 0;
}
void create_list(int p){
  struct node *newnode,*temp;
   int i, data;
   head=(struct node*) malloc (sizeof(structnode));
   printf ("Enter data");
   scanf ("%d", &data);
   head→data=data;
   head→next=NULL;
   temp=head;
   for(i=2; i<=P; i++)
  newnode=(structnode*) malloc(sizeof(structnode
   printf ("Enter data");
   scanf ("%d", &data);
   newnode→data=data;
   newnode→next=NULL;
   temp→next=newnode;
   temp=temp→next;
}
temp→next=NULL;
}
void insert (int n, int data){
    struct node*temp, *newnode;
    int i;
   newnode=(structnode*) malloc(sizeof(structno
   newnode→data=data;
```

```
newnode->next=NULL;
temp=head;
for(i=2;i<=n-1;i++){
    temp=temp->next;
}
newnode->next=temp->next;
temp->next=newnode;
temp=temp->next;
printf("NODE INSERTED SUCCESSFULLY");
}
void delete(int k){
  struct node *temp;
  int i;
  temp=head;
  for(i=2;i<=k-1;i++){
      temp=temp->next;
  }
  temp->next=(temp->next)->next;
  printf("NODE DELETED SUCCESSFULLY");
}
void traverse(){
    struct node *temp;
    temp=head;
    while(temp!=NULL){
        printf("%d\t",temp->data);
        temp=temp->next;
}
```

INPUT:- Enter the number of nodes -7
enter the position to insert 5
enter the data of the inserted node 45
enter the position of node to delete 3
enter data-1
enter the data-2
enter the data-3
enter the data-4
enter the data-5
enter the data-6
enter the data-7
NODE INSERTED SUCCESSFULLY
NODE DELETED SUCCESSFULLY

1 2 4 45 5 6 7

2)

```
#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 * new=(struct node*)malloc(sizeof(struct node))
new→data=data;
new→next = *head;
*head = new;
}
struct node* 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->neaet=b;
    }
  }
   return dummy.neaet;
}
- int keys[] = {1,2,3,4,5,6,7};
  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[j]);
  for (int i=n-2; i>=0; i=i-2){
     push (&b; key[j]);
  struct node *head = merge (a,b);
  printlist(head);
}
```

3)

```
#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+ =arr[h];
        h++;
    if (sum==s){
```

```
    printf(" found");
    return arr[h];
    else {
        sum -= arr[l];
    }
}
int main (void) {
int arr[] = {2,6,0,9,7,3}
 int s = 15;
 int n = size of (arr)/ size of (arr[0]);
 find (arr, n, s);
 return 0;
}.
```

4)

```
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node* next;
};
void printrev(structnode*head){
 if (head==NULL){
    return;
    print rev(head→next);
    printf("%d", head→data);
 void push(struct node* headrev, char new) {
```

```
struct node* node_new = (structnode*) malloc(sizeof(struct
node_new->data = new;
node_new->next = (head*_ref);
(*head_ref) = node_new;
}
int main(){
    struct node* head=NULL;
    push (&head, 4);
    push (&head, 3);
    push (&head, 2);
    printf new(head), print alternative(head);
    return 0;
}
void printalternative (structnode*head){
int count = 0;
while (head != NULL){
    if (count%2==0){
        count << head->data << " ";
        count++
    }   head=head->next;
```

5) a) array contains similar data elements where as li
list can contain different data types.

b) In array variables are recognised by indeces an
can get any element at any position. whereas lin
list we must go from first.

b)
```
#include <stdio.h>
#include <stdlib.h>
int len(int a[]){
  int i=0; an=0.
  while(i)
  {
    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("\n the elements of second array");
```

```
for(int i=0;i<len(b);i++) {
    printf("%d", b[i]);
}
}
int main() {
    int a[10]={1,2,3}, b[10]={4,5,6};
    changinglist(a,b);
}
```

LAB PROGRAMS

DFS

```
#include <stdio.h>
int G[10][10], visited[10], n;
void main() {
    int i, j;
    printf("Enter number of vertices");
    scanf("%d", &n);
    printf("Enter adjecenty matrix 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];
}
```

```
void DFS(int i){
    int j;
    Printf("\n%d",i);
    visited[i]=1;
    for(j=0;j<n;j++){
    if (visitied[j]&&G[i][j]==1)
      DFS(j);
}
```

BFS

```
#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("Enter number of vertices");
    scanf("%d",&n);
```

```
for(i=1; i<=n;i++){
    q[i]=0
    visited[i]=0;
}
Printf("Enter graph data in matrix");
for(i=1; i<=n; i++){
    for(j=1; j<=n; j++) {
        scanf ("%d", &a[i][j]);
    }
}
Printf ("Enter sorting vertices");
scanf ("%d", &v);
BFS(v);
Printf ("The node which are reachable are ");
for (i=1; i<=n; i++){
    if(visited[i]) {
        Printf ("%d\t", i);
    else {
        printf ("BFS not possible");
    }
}
```