C Assignment - 4

J. Vamsi Krishna
AP19110010366
CSE-F

1) Write a C program for depth first search (DFS) using array.

Program:-

```
#include<stdio.h>
 void DFS(int);
 int G[10][10], visited[10], n;
  Void main()
{
    int i,j;
    printf(" Enter the number of vertices");
     Scanf ("%d, &n);
     printf("\n Enter the adjencey matrix of 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 (!visited [j] && G[i][j]==1)
       DFS(j); }
```

output :-

```
Enter number of vertices: 3
Enter adjecency matrix of the graph:
 1  2 1
 1  3 1
 1  4 1
```

2) Write a C program for breath first Search (BFS) using array

program:

```
#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 the no.of vertices");
   scanf("%d"; &n);
for (i=1 ; i<=n ;i++ )
{ q [i]= 0;
  visited [i]=0;
}
 Printf(" \n Enter graph data in matrix form \n");
  for (i=1; i<=n; i++)
 { for (j=1; j<=n; j++)
 { sconf("%d", &a[i][j]);
 }
}
 printf (" \n Enter the starting vertex");
  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", i);
   else
  {
     printf(" \n BFs is not possible. Not all nodes
      are reachable");
     break;
 }
}
}
```

~~output:~~

3) Write a program to insert and delete an element at nth and kth position in a linked list where n and k is taken from user.

Program:-

```
#include<stdio.h>
#include <stdlib.h>
struct node
{
  struct node *next;
};
struct node *curr, *temp
 void input(struct node *)
 void delete(struct node *)
  void main(void)
 {
   struct node *s;
   int n;
   s=Null;
   do
 {
  printf("Enter the elements to insert:\n");
   printf("2. Delete \n");
   printf("3. Exit \n");
   printf("Enter the choice:");
    scanf("%d",&n);
      switch(n)
    {
     case1: input(s);
     break;
  case2: delete(s);
     break;
  } while(n!= 3)

}
Void input (struct node *z)
 {
     int  pos, c=1
    curr=z;
   printf ("Enter the element to be inserted");
   Scanf ("%d", &pos);
      while(curr->next != Null)
    {
       c++;
      if(c==pos)
   {
    temp=(struct node *)malloc(size of (struct node));
    Printf ("Enter the numbers");
      scanf ("%d", &temp->n);
         temp->next =curr->next
          curr->next = temp;
          break;
      }
    }
 }
 void  delete (struct  node *z);
 {
     int  pos, c=1;
       curr=z;
 Printf ("Enter the elements to be delete");
Scanf ("%d", & poss);
 while(curr ->next != Null )
 {
    c++;
    if (c==pos)
{   temp(current ->next);
    curr -> next = curr -> next ->next;
     free(temp)
  }
curr=curr -> next;
}
void merge (struct node *p, struct node *q)
{ struct node *p_curr =p, *q_curr=*q;
  struct node* p_next, *q_next;
 while (p_curr=Null && q_curr!=Null)
{  p_next=p_curr ->next;
   q_next = q_curr->next;
   q_curr -> next =p_next;
    p_curr ->next=q_curr;
    p_curr =p_next;
      q_curr =q_next;
 }
  *q= q_curr
}
int main()
{
```

```
struct node *P = Null, *q = Null;
Push (&P, 1);
Push (&P, 2);
Push (&P, 3);
Printf ("First link list: \n");
Printf list (P);
Push(&q, 4);
Push (&q, 5);
Push (&q, 6);
Printf(" Second link list is: \n");
Print list (q);
merge ( P, &q);
Printf (" Modified first link list = \n");
Print list (P);
Printf ( " Modified second linklist = \n");
Print list (q);
return 0;
}
```

4) Construct a new linklist 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} in the new list we should have {1, 2, 3, 4, 5, 6}

Program:-

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <assest.h>
struct node
{ int data;
  struct node *next;
};
 void move node(struct node **x ; struct
        node * * y);
 struct node * sorted merge(struct node *a,
        struct node * b)
 { struct node dummy
      struct node * tail=&dummy;
      dummy.next=NULL;
      while(1)
   {  if (a==NULL)
    {
        * if =new node->next;
        new node->next=*x;
           *x = newnode;
  }
  void push (struct node ** head_ref, int
        new_data)
   { struct node* new_node=(struct node*)
     malloc size of (structnode));
     new_node->data=new_data;
    new_node->next=(*head_ref);
(*head-ref)=new-node;
}
void   point list (struct node *node)
{
   while (node != Null)
  { printf ("%d", node->data);
    node= node->next;
 }
}
      tail->next=b;
       break;
 }
 else if (b==Null)
  { tail-> Next=a;
     break;
 }
 if (a->data <= b->data)
  {
    move node((tail)->next, &a);
  }
 else
   {
     move node (&(tail)->next, &b);
   }
 tail=tail->next;
 }
 return(dummy next);
 }
 void movenode (struct node **x, struct node
        **y)
 {   struct node *new node = *y;
```

```
assert (new node != NULL);
int main()
{
    struct node* res = null;
    struct node* a = NULL;
    struct node* b = NULL;
  push(&a, 1);
  Push(&a, 2);
  Push(&a, 3);
  Push(&a, 4);
  Push(&a, 5);
  Push(&a, 6);
 res = sorted merge (a, b);
 printf ("Merge link list is: \n");
 print list (res);
 return 0;
}
```

5) Find all the elements in the stack whose sum is equal to k (where k is given from user)

```
#include <stdio.h>
 int s1[10], top1 = -1, S2[10], top2 = -1;
int s1 empty()
{ if (top1 == -1)
     return 1;

   else
       return 0;
}
int s1 top()
{ return s1[top1];
}
int s1 top()
{
     top1--;
}
int s1 push(int x)
{ s1[top1++]=x;
}
int s2 empty()
{
    if (top2==-1)
        return 1;
    else
        return 0;
}
int s2 top()
{
    return s2[top2];
}
int s2 pop()
{ top2--;
}
int s2 push (int x)
{
    s2(top2++)=x;
}
```

```
int  sum (int  k)
{ int x;
   while (s1 empty() != 1)
  {   x = s1 top();
      s1 pop;
  while (s1 empty() != 1)
  {  if (x + s1 top() = k)
     { printf( %d , %d ) \n", x, s1 top();
     }
      S2 push(s1 top());
       S1 pop();
   }
   while (S2 empty () != 1)
   {
      S1 push (S2 top());
      S2 pop();
   }
 }
int  main()
{
   int  n, i, e, k;
printf ("Enter the no.of elements of stack:")
 scanf ( "%d", &n);
 for (i=0; i<n; i++)
 {
    scanf("%d", &e);
    s, push (e);
}
printf("Enter the value of constant sum:\n");
 scanf("%d", &k);
printf("The combinations whose sum is equal
 to k is:\n");
sum(k);
}
```

6) Write a c program to print the elements in a queue in (i) reverse order (ii) alternate order.

Program:-

```
(i)#include <stdio.h>
#include" stack.h"
#include "QQ.h"
int main()
{ int n, arr[20], i, j=0;
  struct stack s;
  init stack(&s);
   printf("Enter no");
    scanf("%d", &n);
  for(i=0;i<n;i++)
  { printf("Enter values:");
     scanf("%d", &arr[i]);
```

```
}
for(i=0;i<n;i++)
{  insert (arr[i]);
}
  while (j!=n)
  {
    push(&s, def());
    j++;
}
  printf ("Reverse is:");
    while (stop!=1)
   {  prinf ("%d", pop(&s));
   }
    printf ("\n");
 return 0;
}
(ii) #include <stdio.h>
   # include <stdlib.h>
     struct node {
       int data
      struct Node *next;
   }
  void print nodes(struct node *head)
   {  int count =0
      while (head != Null {
      (count % 2 == 0{
```

```
printf ( " %d", head->data );
}
  count ++;
  head = head->next;
}
}
void push ( struct Node * * head - ref, int new -
      data )
{ struct hode* new -node= (struct hode) malloc
   (sizeof  (struct node));
  new-node -> data= hew-data;
    new - hode->next = ( *head - ref);
 (*head -ref ) = hew-node;
}
 int main( )
 {  struct  node*  head = NULL;
      push (&head , 12);
      push(& head, 29);
      push(& head, 11);
      push (& head, 23);
      push (& head, 8);
       print  hode  (head);
 return 0;
}
```

7) (i) How array is different from the link list
(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.

Ans (i)
Array:- It is collection of elements having same datatype with a common name
linked list:-
It is an ordered collection of elements which are connected by links or pointers
→ Array and linked list are mainly differ by size, memory allocation, access etc.

→ In size parameter Array is Fixed, once declared cannot be changed. But whereas linked list it is a Variable, can be changed during time to time.

→ In memory allocation parameter array requires continous blocks of memory. But in linked list it is "random memory can be Allocated".

→ ~~In memory~~ ~~a~~ In searching arry have binary search and linear seach but linked list have only linear search.

(ii)
```
#include <stdio.h>
 #include <stdlib.h>
struct hode
{ int data;
 struct node *next;
}
void push (struct node * * head-ref);
{
 Struct hod ** new-node = (struct hode *)
malloc (sizeof (struct hode));
 new-hode->data = new-data;
 New-hode->next = ( &head-ref);
  (*head-ref) = new-hode;
}
void print list (struct hode *head)
 {
  struct node *temp =head;
   while(temp! =NUll)
  { printf("%d", temp->data);
    temp = temp->next;
  }
  printf("\n");
}
```