DSA Assignment

J. Nano; Maharandi
AP21110010390
CSE - F

1)
```
#include <stdio.h>
#include <stdlib.h>
 struct node
 {
 struct node * next ;
 };
 struct node *cur,*temp;
 void input (struct node*)
 void delete (struct node*)
 void main (void)
 {
 struct node *s;
 int n;
 s=NULL;
 do
 {
 printf ("enter the element to insert ;\n;");
 Printf ("2. Delete\n");
 printf ("3.exit\n");
 printf ("enter the choice:");
 scanf ("%d", &n);
 switch (n)
 {
  case 1: input (s);
          break;
```

```
case 3: delete(s);
        break;
      } while(n:=3)
}
void input (struct node *z)
{
 int pos, c=1
     coss = z;
printf("enter the element to be inserted");
scanf("%d", &pos);
     while (coss -> next != NULL)
 {
  c++;
 if (c==pos)
 {temp = (struct node*).malloc.(size of struct node));
printf("enter the numbers");
scanf("%d", &temp->n);
   temp -> next = coss -> next
    coss -> next = temp;
   break;
   }
  }
}
 void delete (struct node *z)
 { int pos, c=1;
```

```
curr = 2;
printf("enter the element to be delete:");
scanf("%d", &pos);
while (curr->next != NULL)
{
 c++;
 if (c == pos)
{
  temp = current->next;
  curr->next = cur->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 = p-next;
  q-curr = q-next;
 }
```

```
*q = q->next
}
int main()
{
struct node *p = NULL, *q = NULL;
push(&p, 1);
push(&p, 2);
push(&p, 3);
printf("first linked list:\n");
print list(p);
push(&q, 4);
push(&q, 5);
push(&q, 6);
printf("second linked list:\n");
print (list(p));
printf("modified second linked list=\n");
print list(q);
return 0;
}
```

2.
```
#include <stdio.h>
#include <stdlib.h>
#include <assert.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;
dummy.next = NULL;
while (1)
{
if (a==NULL)
{
*y = newnode -> next;
newnode -> nex = *x;
*x = newnode;
}
void push (struct node ** head-ref, int new-data)
{
struct node * new-node = (struct node *) malloc size of
                         (struct node));
```

```
new-node -> data = new-data;
new-node -> next = (*head-ref);
   (*head-ref) = new-node;
}
void point list (struct node *node)
{
 while (node = NULL)
 {
  pointf ("%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 (&(*a) -> next), &a);
 }
 else
 {
  move node (&tail) -> next, &b);
 }
  tail = tail -> next;
```

```
return (dummy.next);
}
void movenode(struct node **, struct node **);
{
struct node *newnode = *y;
assert(newnode != NULL);
int main()
{
struct node *res = null;
struct node *a = NULL;
struct node *b = NULL;
Push(&a, 1);
Push(&b, 2);
Push(&a, 3);
Push(&b, 4);
Push(&a, 5);
Push(&b, 6);
res = sorted merge(a,b);
printf("merge linked list is :\n");
Print list(res);
return 0;
}
```

3) #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(top);
}
int S1POP()
{
top1--;
}
int S1 Push (int x)
{
S1(++top1)=x;
}
int S2 empty()
{
if (top2==-1)
  return 1;
 else
 return 0;
}
int S2top()
{
return S2[top2];
```

```
int main()
{
  int n,i,e,t;
printf("enter the no. of element of stack:\n");
scanf("%d",&n);
for(i=0;i<n;i++)
{
 scanf("%d",&e);
 S1.Push(e);
}
printf("enter the value of const. sum:\n");
scanf("%d",&k);
printf("The combinations whose sum is equal to k is:\n");
 Sum(k);
}
```

4) i)

```
#include<stdio.h>
#include "stak.h"
#include "QQ.h"
 int main()
{ int.n,arr[20],i,j;
 struct stack S;
 Initstack(&S);
 printf("Enter no:");
```

```
int S2POP()
{
top2--;
}
int S2 Push(int x)
{
S2(1+top2)=x;
}
int sum(int k)
{
int x;
while(S1.empty()!=1)
{
x=S1.top();
S1.POP();
while (S1.empty()!=1)
{
~~x=S1.top~~ if(x+S1.top()=k)
{
printf("%d,%d)\n", x1, S1.top());
}
S2 Push (S1.top());
S1.POP();
}
while (S2.empty()!=1)
{
S1.Push(S2.top());
S2.POP();
}
}
}
```

```
scanf("%d",&n);
for(i=0;i<n;i++)
{
 print("enter values:");
 scanf("%d",&arr[i]);
}
for(i=0;i<n;i++)
{
 for(i=0;i<n,i++)
{insert(arr[i]);
}
while(j)=n)
{
 push(&s)del(j);
 j++;
}print("Reverse:s");
while(stop!=-1)
{
 printf("%d",Pop(&s));
}
 printf("\n");
 return 0;
}
```

ii)
```
#include<stdio.h>
#include<stdlib.h>
struct node.
{
int data;
struct node *next;
};
void print note (struct node* head)
{
 int count=0;
while (head!=NULL){
if (count % 2==0){
Printf ("%d", head->data);
}
count ++;
head= head->next;
 }
}
void push (struct node* head-ref, int new-data)
{
struct node *new-node = (struct node*) malloc
                        (sizeof struct node);
new-node->data = new-data;
new-node -> next = (*head-ref);
(*head-ref)=new-node;
```

```
}
int main()
{
struct node * head = NULL;
}
```

5) i) The major diff b/w Array & linked lists regards to their structure, Arrays are index based data. Structure where each element associated with an index on the other hand. linked list roles on refrence to the previous and next element.

ii)

```
#include <stdio.h>
#include < std lib.h>
 struct node
 {int data;
  struct node *next;
 }
 void push (struct node** head-ref)
{
struct node * new-code = (struct node*) malloc
                          (size of struct node));
new-node -> data = new-data;
new-node -> next = (&head-ref);
```

```
(*head_ref) = new_node;
}
void printList(struct node *head)
{
struct node * temp = head;
while (node * temp != NULL)
{
printf("%d", temp->data);
temp = temp->next;
}
printf("\n");
}
```