ASSIGNMENT - 4

O. ARAVIND
AP19110010523

1)

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

Void ins ( node* , int , int )
int size = 0 ;
struct node
{
  int data ;
  struct node * next;
}

Node * get node (int data)
{
 Node * new node = (struct node *) malloc (new node);
 new node → data = data;
 new node → 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 print (struct node * head)
  {
    while (head != Null)
     {
       printf ("%d", head → data);
          head = head → next;
     }
     printf ("\n")
}
 void del (struct node *head ref, int pos)
{
 if (head_of = =Null)
    return;
   temp = head_ref;
if (pos =0)
{
* head ref = 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, 7);
  push (& head, 8);
  push (& head, 6);
  ins (& head, 7, 5);
  del (& head, 4);
  print list (head);
  return (0);
}
```

2)

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
int data;
struct node * next;
};
void print list (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 (size of (struct node));
 new -> data = data;
 new -> next = * head;
    * head = new;
}
stauct node * merge (struct node *a , struct node * b)
{
Strud node dumy;
struct node * tail = dumy
dumy . next = Null;
```

```
#include <stdio.h>
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;
}
void main()
{
int keys[] = {1,2,3,4,5,6,7};
int n = sizeof(keys)/sizeof(keys[0]);
struct node *a = Null, *b = Null;
for (int i = n-1; i >= 0; i = i-2)
```

```
        push (aa, keys[i]);
    for (int i=n-2 ; i>=0 ; i=i-2)
        push (ab, key[i]);

    struct node * head = merge (a,b);
    print list (head);
}
```

3)

```
# include <stdio.h>
void find (int ar[], int n, int s)
{
  int sum = 0;
  int l=0, h=0;
  for (l=0; l<n; l++)
   {
    while (sum <s && h<n)
     {
      sum += ar[h];
      h++;
    if (sum == s)
    {
     printf ("found");
    return;
    }
```

```
        Sum == ar[i];
    }
}
int main (void)
{
 int ar[] = { 2, 6, 0, 9, 7, 3}
 int   s= 15;
 int  n = size of (ar) / size of (ar[0]);
 find (ar, n, s);
 return 0;
}
```

4)

```
# 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);
 printf ("%d", head → data);
void push (struct node *head ref, 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, 1);
  push (&head, 3);
  push (&head, 2);
  print rec(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)

```
#include <stdio.h>
int main()
{
 int a1[100], a2[100];
 int i, x, pos, n=10;
 for (i=0; i<10; i++)
 scanf ("%d", &a1[i]);
 scanf ("%d", &a2[i]);
 for (i=0; i<n; i++)
  printf ("%d", a[i])
  printf ("%d", a[2])

x = a2[0];
 pos = 1;
 n++;
for (i=n; i>=pos; i--)
   a1[i] = a1[i-1];
   a[pos-1] = x;
 for [i=0; i<n; i++)
   printf ("%d", a1[i]);
 for (i=1; i<n, i++)
    printf ("%d", a2[i]);
return 0;
```