DSA ASSIGNMENT-4

G. Sumanth Varma
AP19110010208
CSE - G

1.
```
# 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");
  }
print list (head);
 return (0);
}
```

② #construct a new linked list by merging alternate nodes of 2 lists

```
#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");
}
void push(struct node** head, int data)
{
 struct ~~s~~node* newnode = (structnode*) malloc(sizeof(structnode));
 newnode —> data = data ;
 newnode —> next = *head;
 *head = newnode ;
}
struct node *shuffleMerge(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 -> next = b;
    tail = b
   b = b -> next ;
   }
 }
 return dummy. next ;
 }
 int main(void)
 {
   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(&a , keys[ i ]);
 for (int i = n - 2 ; i >= 0 ; i = i -2)
        push(&b, keys [ i ]);
 printf ("First List : ");
 printList (a) ;
 printf ("Second List : ")
                          ;
 printList (b);
 struct node *head = ShuffleMerge (a , b);
 printf ("After Merge: ")   ;
 printList (head);
 return 0;
}
```

```
③ #include <stdio.h>
  int top = -1;
   int x ;
  char stack [100];
  void push (int x) ;

 char pop ( );
 int main ( )
 {
  int i, n, a, t, k, f, sum = 0,   count = 1 ;
 printf ( "enter number of elements in stack ) ;
 scanf ("%d", &n) ;
 for (i = 0; i<n ; i++) {
 printf ( "enter next element");
 scanf ( "%d", &a);
 ~~for~~ ~~(~~ push (a);
  }
  printf ("enter sum to be checked);
  scanf (" %d , & k) ;
  for (i=0;    i<n ; i++)
  {  t = pop ( );
   sum + = t ;
   count ++ = 1 ;
   if (sum = = k) {
  for (int j = 0;  j < count ; j++)
  printf (" %d", stack [i]);
   f = 1 ;
  break ;
   }
  push (t) ;
  }
  if (f ! = 1)
  printf (" The elements in stack do not add uptothe sum);
void push(int x)
{
 if (top = = 99)
  {
   printf(" stack is full !");
  return ;
  }
  top = top+1 ;
 stack [top] = x;
 }
 char pop()
  {
   if (stack [pop] = = -1)
  {
  printf(" stack is empty !");
 return 0)
   }
   x = stack  [top];
   top = top -1
    return x;
    }
```

Scanned by CamScanner

```
(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 ->next) ;
       void push (struct node* head rev ,char new)
     {
      struct node* node _ new= (struct node *) malloc size
                                         of (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, 3);
    push (&head, 2);
     print new(head); print alternate (head);
     return 0;
   }
   void print alternate (struct node*hea)
    {
    int count =0
```

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

Scanned by CamScanner

5

i) The major difference between Array and Linked list regards to their structure. Arrays are index based data structure where each element associated with an index. Linked list relies on references where each node consists of data and the references to previous & next element.

ii)
```
#include <stdio.h>
# include <stdlib.h>
int len (int a [ ])
{
 int i=0, an=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 (" The element of 1st array: ");
 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];
}
print f ("the elements of second array:");
for (int i=0;  i<len(b);  i++)
{
  print f (" %d", b[i]);
  }
}
int main()
{
int a[10]= {1,2,3}, b[10]={1,5,6};
changing list (a,b);
}
void del (struct node *head, int pos){
if (head-ref == null)
return;
 temp= head-ref;
if (pos =0)
{
 * head-ref = temp -> next;
 free (temp);
return;
for(int i=0; temp != NULL & T 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);
 ~~push~~ ins (&head, 7,5);
  del(&head,4);
}
```