Souvik Mandal
AP19110010485
CSE - F

Assignment - 4

1)
```
# include < stdio .h>
# include < malloc . h>
# include < stdlib. h>
   struct node {
            int value ;
            struct ~~sto~~ node * next ;
   }
   void insert ( ) ;
   void delete ( ) ;
   void display ( ) ;
   int count ( ) ;
   type def struct node Data - Node .
   Data - Node * head - node , * first - node , * temp - node = 0 ;
        * prev - node , next - node ;
   int data ;
   int main ( ) {
    int option = 0 ;
   printf (" singly linked list Example . all operations \n ") ;
   While ( option < 5) {
   printf (" \n otions \n") ;
   printf (" 1. Insert into linked list \n") ;
   printf (" 2. Delete from linked list \n") ;
   printf (" 3. Display linked list \n") ;
   printf (" 4. Count linked list \n") ;
   printf (" others : exit() \n") ;
   printf (" Enter your option:") ;
scanf(" %d", & option);
switch ( option ) ;
 case I ;
  insert ( ) ;
   break ;
 case 2 ;
  delete ( ) ;
  break ;
 case 3 ,
  display ( ) ,
  break ( ) ;
 case 4 ,
  count ) ;
  break ;
}
}
return 0 ;
}
void insert ( ) {
printf ("\n Enter elements for inserting in linked list \n");
scanf(" %d", & data);
 temp - node = (Data_Mode*) malloc ( size of ( Dak - Node))
    temp - node → value = data ;

   if ( first - node = = 0) {
         first - node = temp - node ;
      }
    else {
     head - node → next = temp - node
    }
```

```
temp - node -> next = 0 .
head - node = temp - node ;
    flush ( stdin) ;
  }

void delete ( )
 {
  int count value, pos, i = 0 ;
     count value : count ( ) ;
      temp - node = first - node ;
    printf (" \n display linked list :\n") ;
   printf (" \n Enter position for deleting element :\n") .
      if ( pos > 0 && pos <= count value) {
        if (pos == 1) {
          temp - node = temp - node -> next ;
        first - node = temp - node ;
   printf (" \n delete successfully \n") ;
   }
   else {
    while ( temp - node != 0) {
     if ( i == (pos - 1)) {
      prev - node -> next = temp - node -> next ;
          if (i == 0( count value - 1)) :
      {
        head _ node = prev - node ;
      }
      printf (" \n Deleted successfully \n \n") ;
      break ;
}
else
{
 i++;
  prev - node = temp - node;
  temp - node = temp - node -> next;
  }
  }
  }
 } else
   printf ("\n Invalid position \n");
 }

void display () {
      int count = 0;
    temp - node = first - node;
    printf ("\n display linked list : \n");
    while ( temp - node != 0) {
    printf (" #%d#", temp - node -> value);
       count ++;
    temp - node = temp - node -> next;
    }
    printf ("\n no of items in linked list : %d\n", count);
 }

~~int count () {~~
~~   int count = 0;~~
~~   temp - node = first - node;~~
~~   while ( temp - node != 0) {~~
~~     count ++;~~
```

~~temp_node = temp-node → next~~

}

~~printf ("\n no of items in linked list : %d \n".~~

return count ;

}

2)
```
# include <stdio.h>
# include <stdlib.h>
struct node
{
 int data ;
 struct Node * next;
};
void printList (~~struct~~ 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 node = (struct node*) malloc (
    sizeof (struct node));
```

```
New Node -> data = data ;
New node -> next = * head ;
* head = new node ;
}
struct node * shuffle merge (struct node * a, struct
                                    node * b)
    {
      struct node dummy ;
      struct Node * tail ; & dumy ;
      dummy . next = NULL ;
       while (1)
     {
       if ( a = = null)
       {
        tail -> next = b ;
              break ;
       }
       else if (b = NULL)
       {
        tail -> next = a ;
           break ;
     }
     else
     {
      tail -> next = a ;
           tain = a ;
         a = a -> next
         tail -> next = b ;
            tail = b ;
        b = b → next ;
    }
  }
  return deemy . next ;
  }
  int main( void )
  {
   int keys [ ] = { 1, 2, 3, 4, 5, 6, 0 } ;
    int n = sizeof (key) / sizeof (keys [0]) ;
    struct node * a = NULL, * b = NULL ;
    for ( int i = n - 1 ; i >= 0 ; i = i - 1) ;
        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 = shuffle merge ( a, b ) ;
   printf ( "after merge :") ;
     printlist ( head ) ;
     return 0 ;
   }
```

Input → output

First list : 1 → 3 → 5 → 7 → NULL

second list : 2 → 4 → 6 → NULL

After merge : 1 → 2 → 3 → 4 – 5 – 6 → 7.

3)
```
# include <stdio.h>
   int top =1 ;
     int x ;
   char stack [100];
   void push ( int x);
   char pop ( ) ;
   int main ( )
   {
    int i, n, a, x, u, t, sum =0, count = 1 ;
   printf (" Enter the number of elements in the stack") ;
   scanf ("%d", & n) ;
    for( i =0; i<n ; i++)
    {
    printf (" Enter next elements");
    scanf ("%d", & a) ;
     Push (a) ;
      }
     printf (" Enter the sum to be check ") ;
    scanf (" %d ", & n) ;
    for (i =0 ; i<n ; i++)
    {
```

```
i = pop () ;
Sum + = t ;
Count + = 1 ;
  if ( sum = = n) {
 for ( int j = 0 ; j < cout, j++)
 printf ( " %d ", stack [j]) ;
    f = 1 ;
    break ;
 }
  push (t) ;
 }
  if ( f ! = 1)
printf ( " The elements in the stack dont add up to the
        sum ") ;
}
 void push ( int x)
  {
   if ( top = = 99)
   {
   printf (" \n stack is full !) \n ") ;
  return ;
 }
  top = top +1 ;
  stack [ top ] = x ;
  }
```

```
char pop ()
{
 if (stack[top] = = -1);
 {
  printf(" \n stack Empty !!!\n");
  return 0;
}
  x = stack [ top ];
   top = top -1;
    return x;
}
```

output

```
 Enter number of elements in stack 3.
  Enter element 4.
  Enter element 5
 Enter element 4.
 Enter the sum to be checked 30.
  The elements in the stack do not equal to sum.
```

4)

```
 # include <stdio.h>
 # define size 10;
   void insert (int);
   void delete ();
   int queue [10], f = -1, r = -1;
    void main () {
     int value, choice;
```

```
while (1)
{
 printf ("\n\n *** Menu *** \n");
 printf ("1. Insert \n .2. deletion \n 3. Print Reverse \n
      4. print alternate \n 5. exit");
 printf ("Enter your choice");
 switch ( choice) {
   case 1: printf ("Enter the value to be inserted:");
        scanf ("%d", &value);
       insert ( value);
       break;
   case 2: delete ();
   break;
   case 3:
       printf ("The reversed queue is ");
       for (int i = size; i>=0; i--);
     {
      if (queue [i]==0)
          continue;
          printf("%d", queue [i]);
     }
      break;
      case 4:
       printf ("alternate element of the queue are");
       for (int i =0; i< size; i+=2)
       {
```

```
if ( queue [i] = = 0)
      continue ;
      printf ("%d", queue [i]);
   }
   break ;
  case 5 ; exit (0)
  default : printf ("\n wrong select , Try again");
   }
 }
)

 void insert ( int value) {
  if (( f = = 0 && r = = size - 1) || f = = r + 1)
   printf (" Queue is full, Insertion is not possible ");
    else {
      if (f = = 1)
       f = 0;
       r = (r + 1) % size ;
      queue [r] = value ,
      printf ("\n insertion success");
      }
     }
   void deletion() {
```

```
if (f == -1) {
printf("\n Queue is empty, deletion is not possible)".
else {
printf("\n deleted : %d", queue[f]) ;
  f  = (f + 1) % size ;
   if ( f == r)
  f = r = -1
    }
```

5) i) The difference between the array & the linked list regards to their structure. Arrays are index based data structure where each element associated with an index. On the other hand, linked lists relies on references where each node consists of the data & the references to the previous & next element.

ii)
```
# 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 node = (struct node*) malloc
                  ( sizeof struct node);

  new node -> data = data;
  new node -> next = * head;
   * head = new node;

  }
  void move node (struct Node ** dst ref, struct
      Node ** source ref)

  }
  if (* source ref == Null)
      return;
  struct node * new node = * source ref;
 * source ref = ( * source ref ) -> next;
  New node -> next = * dst ref;
    * dst ref = New Node;
```

```
}
int main ( void)
{
 int keys [ ] = {1, 2, 3}
int n = size of keys ( ) / size of keys [0] ) ;
  struct node * a = Null ;
   for ( int i =0; i< n ; i --)
    push ( &a ; keys [ i ]) ;
   struct node * b = Null ;
  for ( int i = 0; i< n ; i ++)
   push ( &b; 2 * key [i]) ;
  move node ( &a, &b) ;
 printf ("first list") ;
 printlist (a) ;
 printf (" second List :") ;
  printlist (b) ;
  return 0 ;
 }
```

Output -display.

```
First list : 6 → 4 → 2 → 3 → Null
second list : 4 → 2 → NULL
```