Assignment

K. Vamshi Vardhan
AP19110010325
CSE - F

```
#Include <stdio.h>
#Include <malloc.h>
#Include <stdlib.h>
  Struct node {
  int value
 Struct node *next.
  void Insert ()
  void display ()
  void delete ()
  Int count ()
  typedef 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 options\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 ("enters your option:")
      Scanf ("%d" & option):
      Switch (option)
      case 1:
```

```
Insert()
  break.
case 2:
   delete()
    break
case 2:
     display()
     break
case 3:
     count()
     break;
     default:
      break;
      return 0.
    void insert()
Printf("\n Enter Element for Insert linked list :\n")
Scanf("%d", &data);
temp_node = (DATA_NODE *) malloc (size of (DATA_NODE)
temp-node->value = data
if (first_node == 0)
 first_node -> next = temp-node
 temp-node -> next = 0
   head_node = temp-node
   fflush (stdin)
   void delete()
  int countvalue, pos i = 0
  countvalue = count()
  temp_node = first node
    Printf ("n display linked list : \n")
    Printf ("/n Enter position for delete Element : \n")
```

```
scanf ("%d" &pos)
if (pos > 0 && pos <= countvalue)
if (pos == 1)
  temp_node = temp_node -> next
   first_node = temp - node
   Printf ("\n deleted sucessfully \n\n")
   } else {
   while (temp_node != 0)
   if (i == (pos - 1)) {
   Prev_node -> next = temp_node -> next
   if (i == (count value - 1))
  head_node = prev_node.
 Printf ("\n deleted sucessfully \n\n")
   break
   } else {
   i++
   Prev_node = temp_node
   temp node = temp_node -> next
   else
  Printf ("\n Invalid position \n\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" count)
Return count
```

(2)

```
#Include <stdio.h>
# include <stdlib.h>
// Data Structure to store a linked list node
Struct node
int data
  struct node* next

+Helper function to print given listed list
void print list (struct node* head
Struct Node * Ptr = head
  while (Ptr)
    Printf ("%d -> " Ptr -> data)
      Ptr = Ptr -> next
  Printf (" NULL \n").
```

```
// helper function to insert new node in the beginning of
the linked list void push (struct Node * -> head, int data)
{ Struct Node* New Node = (struct Node *) malloc (size of
(Struct Node))
  New Node -> data = data
  New node -> data = data
  *head = new node
```

// function to construct a linked list by merging alternate nodes of // two given linked lists, using dummy node

```
Struct node* Sytle merge (struct node *a, Struct node* b.)
    Struct node dummy
  Struct node* tail = & dummy
   dummy . next = NULL
while (1)
    // empty list cases
    if (a == NULL)
    {
       tail -> Next = b ;
        break
    }
    else if (b == NULL)
     tail -> next = a
      break
tail -> next = b
     tail = b
     b = b -> next
return dummy . next
```

```
// main method
int main (void)
// input keys
int keys [] = {1,2,3,4,5,6,7}
int n = size of (keys) / size of (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)).
// Print both Linked list
Printf ("first list :")
Print list (a)
Print ("second List :").
Print list (b)
Struct Node* head = shyffle Merge (a,b);
Print ) ("After Merge :")
Print list (head)
return 0.
```

(3)

```
# Include <studio.h>
int top = 1
int n.
char stack [100]
void push (int x)
```

```
void push (int x)
  char pop ()
    int main ()
    int i, n, a, t, k, f, 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 Element")
 Scanf ("%. d " & a)
  Push(a)
Printf ( "Enter the 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[j]
      f = 1
      break
      Push (t)
       if (f ! = 1)
  Printf (" The Elements is 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 is EMPTY !!! \n")
return 0
x = stack [top]
top = top - 1
Return x .
```

(4)

```
#include < stdio.h >
# define SIZE 10
void Iwert (int)
void delete ()
int queue [10] ; f = -1 , r = -1
void main ()
int value, ch. choice
while (1)
Printf. ("\n \n*** MENU ** \n") ;
Print f ("1 : Insertion \n 2. deletion \n
3. Print Reverse \n . 4. Print Alternate \n
5. Exit)
Print ("\n Enter you choice")
* Scanf ("%d" choice)
Switch (choice)
```

```
case 1 : Printf ("Enter the value to be Insert");
    Scanf ("%d", &value);
    Insert (value)
      break
    case 2 : Delete ()
      break
  case 3 :
      Print ("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 Elements of the queue are")
    if (queue[i] == 0)
    continue
    Print ("%d" queue[i])
    break
    case 5 : exit (0)
    default : Print ("In wrong Selection [!] Try again !!! )
void Insert (int value) {
    if ((f == 0 && r == SIZE -1) || f == r+1)
  Printf ("\n queue is full !!! Insertion is not poribible !!!
    Else
      if (f == -1)
    f = 0
    r = (r+1) % SIZE
      queue (r) = value
      Printf ("\n Insertion success !!!")
```

```
void delete() {
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) 1) difference blw array & linked list, 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. On the other hand, Linked list relies on References where each node consists of the data and the Reference to the previous and Next Element

5) 2)
```
# include <stdio.h>
# Include <stdlib.h>
// data structure to store a linked list node
  struct node
    int data;
    struct node* node next
Helper function to print given linked list
  void print list (struct node * head)
      struct node* ptr = head.
        while (ptr)
      Printf ("%d -> ", ptr -> data)
      Ptr = Ptr -> next.
```

```
ptr = ptr -> next
Print ("NULL \n");
// helper function to insert 'new node in the beging of the linked
list void push (struct node** head, int head)
Struct Node* new Node = (struct node*) malloc (size of (struct node));
    new node -> data = data
    new node -> next = *head
*head = new Node
// function take the node from the front of the source, and move
it // to the front of the destination
void move node (struct node** dest Ref, struct Node** source Ref
    // if the source list empty do nothing
    if (*source Ref == NULL)
        Return
Struct Node* new node = *Source Ref; // the front Source node
*Source Ref = (*Source Ref) -> next // advance the source pointer
    new node -> next = *destRef ; // Link the old dest off the
new node & *dest Ref = new Node; // move dest to point to
the new method
    // main method
    int main (void)
        // input keys
        int keys [] = {1,2,3}
    int n = sized of (keys) / size of (keys [0])
```

```
// construct first linked list
  Struct node *a = NULL
  for (int i=n-1 ; i>=0 ;i--)
Push (&a , keys [i])
// construct second linked list
  Struct Node *b = NULL
  for (int i=0 ; i<n ,i++)
  Push (&b, 2* keys (i))
// move front node of the b and move it to the front of
the a move Node (&a, &b)
  // Print both List
    Printf ("First List :");
      Print List (a);
Printf ("Second List :")
  Print List (b)
  return (0)
```