Assignment -4

K.N.V.S.S.NITIN
AP19110010484
CSE - F.

1) Write a program to insert & delete an element at the n^{th} & k^{th} position in a linked list, n & k taken from the user.

⇒

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

  Struct node {.
     int value ;
     Struct 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 -dll 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(" Enter your option : ");
scanf(" %d ", & option);
```

```
switch (option) {
case 1;
    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_NODE*) malloc(sizeof(DATA_NODE));
temp_node -> value = data;
if (first_node == 0){.
        first_node = temp_node;
} else {
    head_node -> next = temp_node;
}
temp_node -> nent = 0;
head_node = temp_node;
    flush(stdin);.
}.
```

```
void deleted() {
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 deleted successfully\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 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",
                                        count )
   return count;
}
```

1) Output.

Menu

1) Create.
2) Display
3) Insert a node at specified position.
4) Delete node
5) Exit.

⇒ Enter your choice : 1.
⇒ Enter the data value for the node : 20.

Menu

1. Create.
2. Display.
3. Insert a node at specified position.
4. Delete node
5. Exit.

⇒ Enter your choice : 1.
⇒ Enter the data value for the node : 50.

Menu

1) Create.
2 Display.
3 Insert a node at specified position.
4 Delete node.
5 Exit

⇒ Enter your choice : 3.
⇒ Enter the position for the new node to be inserted.
⇒ Enter the data value of the node : 44.

Menu

1. Create.
2. Display.
3. Insert at specified position.
4. Delete from specified position
5. Exit.

⇒ Enter your choice : 4.
⇒ Enter the position of the node to be deleted : 3
⇒ position not found.

Construct a new linked list by merging alternate nodes of two lists for example in list 1 we have {1, 2, 3} & in list 2 we have {4, 5, 6} in the new list we should have {1, 4, 2, 5, 3, 6}.

```
#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\n");.
}.
void push (struct Node **head, int data).
{.
  struct Node* new node = (struct node*) malloc (
                    size of (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 = & 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(key)/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;
}
```

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 n;.
char stack [100];.
 void push ( int n);
 char pop ();
 int main ().
{ int i, n, a, t, u, 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", &u);
 for (i=0; i<n; i++)
{.
   t = pop ();
   sum + = t;
   count + = 1;
    if (sum == u) {.
for(int j =0 ; j < count, j++)
printf ("%d", stack [j]);
   f = 1;
   break;
}
   push (t);
{ if (f != 1).
```

```
printf("The elements in the stack don't add up to th
        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;.
}.
```

3→ Output

Input - 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) Write a program to print the elements in queue

i) Reverse Order.

ii) Alternate order.

```
#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. Insertion \n 2. Deletion \n 3. Print reverse \n
        4. Print alternate \n 5. Exit \n");
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 elements 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 selection!!, 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;
    }
  }
```

4)

Menu

1. Insertion
2. Deletion
3. Print reverse
4. Print alternate
5. Exit

Enter your choice: 1.
Enter the value to insert: 6.
Insertion success

Menu

1. Insertion
2. Deletion
3. Print reverse
4. Print alternate
5. Exit

Enter your choice: 1.
Enter the value to be insert: 20.

Menu

1. Insertion
2. Deletion
3. Print revere
4. Print alternate
5. Exit.

Enter yow choice: 3.
The reverse queue is 20 6.

Menu

1. Insertion
2. Deletion.
3. Print reverse.
4. Print Alternate.
5. Exit.

Enter your choice: 4.
Alternate elements of queue are: 6.

Menu

1. Insertion.
2. Deletion.
3. Print reverse.
4. Print alternate
5. Exit.

Enter your choice: 5

5)i) How array is different from linked list.

⇒ 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.

5:ii) Program

```
#include <stdio.h>.
#include <stlib.h>.
 Struct node.
{  int data.;
   Struct Node* next;
};
void printflist (Struct Node* head).
     {.
       Struct Node* ptr = head;
         while (ptr).
{.
  printf("%od ->", ptr -> data);
      ptr = ptr -> next;
3.
  printf("NULL\n");
3.
void push (struct node** head, int data)
{.
Struct node* new node = (struct node*) malloc
                                  (size of struct node));
```

```
new node -> data = data;
new Node -> next = *head;
  *head = new Node;
}.

void moveNode (struct Node **destRef, struct
                    Node** sourceRef).
{.
  if (*sourceRef == NULL).
    return;
struct node* newnode = *sourceRef;
*SourceRef = (*SourceRef) -> next;
New Nod -> next = *desRef;
  *destRef = 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: ");
print list (a);
printf ("second list: ");
print list (b);
return 0;
}
```

Input → output;

First list : 6 → 4 → 2 → 3 → NULL.

Second list : 4 → 2 → NULL.