Assignment :- 4

J. Praharshini
AP19110010461
CSE:F

2) construct a new linked list by merging alternative nodes of two lists for example in list 1 we have {1,2,3} and 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 info;
  struct node *link;
};

struct node *create_list(struct node*start);
void display(struct node *start);
struct node *addatbeg(struct node *start, int data);
struct node *addatend(struct node *start, int data);
void combineAlternate(struct node **start1, struct node
                                              **start2);

int main()
{
  struct node *start1 = NULL, *start2 = NULL;
  start1 = create_list(start1);
  start2 = create_list(start2);
  printf("\nFirst ");
  display(start1);
  printf("\nsecond ");
  display(start2);
  combineAlternate(&start1, &start2);
  printf("\ncombined ");
  display(start1);
```

```
    return 0;
}
void combineAlternate (struct node** start1, struct
                                    node** start2)
{
struct node *p1, *p2, *pnext;
p1 = *start1;
p2 = *start2;
if(p1 == NULL)
{
*start1 = *start2;
*start2 = NULL;
 return;
}
if(p2 == NULL)
   return;
while(p1->link != NULL && p2 != NULL)
{
  pnext = p1->link;
  p1->link = p2;
  p1 = pnext;
  pnext = p2->link;
  p2->link = p1;
  p2 = pnext;
}
if(p1->link == NULL)
    p1->link = p2;
  *start2 = NULL;
}
```

```
struct node *create_list(struct node *start)
{
int i,n,data;
printf("\n Enter the number of nodes: ");
scanf("%d",&n);
start = NULL;
if(n==0)
  return start;
printf("\n Enter the Element to be inserted:");
 scanf("%d",&data);
 start = addatbeg(start,data);
for(i=2;i<=n;i++)
{
printf("\n Enter the Element to be inserted:");
 scanf("%d",&data);
 start = addatbeg(start,data);
}
 return start;
}
void display(struct node * start)
{
 struct node *p;
 if(start==NULL)
{
 printf("\nlist is empty\n");
  return;
}
 p = start;
 printf("List is:\n");
```

```
    while (p != NULL)
    {
        printf ("%d ", p->info);
        p = p->link;
    }
    printf("\n\n");
}
struct node *addatbeg(struct node * start, int data)
{
    struct node *tmp;
    tmp = (struct node*) malloc(sizeof(struct node));
    tmp->info = data;
    tmp->link = start;
    start = tmp;
    return start;
}
struct node * addatend(struct node * start, int data)
{
    struct node *p, *tmp;
    tmp = (struct node*)malloc(sizeof(struct node));
    tmp->info = data;
    p = start;
    while (p->link != NULL)
        p = p->link;
    p->link = tmp;
    tmp->link = NULL;
    return start;
}
```

output:

```
Enter the number of nodes: 4
Enter the Elements to be inserted: 1
Enter the Elements to be inserted: 3
Enter the Elements to be inserted: 2
Enter the Elements to be inserted: 4
Enter the number of nodes
Enter the Elements to be inserted: 6
Enter the Elements to be inserted: 5
Enter the Elements to be inserted: 7
First list is:
1 3 2 4
second list is:
6 5 7
combined list is:
1 6 3 5 2 7 4
```

11) Write a program to insert and delete an element at the n^{th} and k^{th} position in a linked list where n and k is taken from user.

```
# 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("\noptions\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;
        default;
        break;
    }
}
 return 0;
}
void insert(){
  printf("\n Enter Elements for Insert 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->next = 0;
 head_node = temp_node;
 fflush(stdin);
}
void delete(){
  int countvalue, pos, i = 0
  countvalue = count();
```

```
temp_node = first_node;
printf ("\nDisplay linked list :\n");
printf ("\nEnter position for Delete Element:\n");
}else{
while(temp_node != 0){
if (i==(pos-1)){
prev_node->next = temp_node->next;
if (i==(countvalue-1))
{
        head_node = prev_node;
    }
printf ("\nDeleted successfully\n\n");
break;
}else{
i++;
prev_node = temp_node;
temp_node = temp_node->next;
}
}
}
}else
printf ("\nInvalid position\n\n");
}
void display() {
int count = 0;
temp_node = first_node;
printf ("\nDisplay linked list :\n");
while (temp_node != 0){
```

```
        printf("#%d#", temp_node->value);
        count++;
        temp_node = temp_node->next;
    }printf("\nNo of items in linked List: %d\n", count);
}intcount(){
    int count = 0;
     temp_node = first_node;
    while (temp_node != 0){
        count++;
      temp_node = temp_node->next;
    }printf("\nNo of items in linked list: %d\n",
            count);
    return count;
}
```

output:

singly linked list Example - dll operations

options

1: Insert into linked list

2: Delete from linked list

3: Display linked list

4: count linked list

others : exit()

Enter your option: 1

Enter Element for Insert linked list : 2

options

1 : Insert into linked list

2: Delete from linked
3: Display linked list
4: count linked list
others: Exit()
Enter your option: 1
Enter Elements for Insert linked list: 3

options:
1: Insert into linked list
2: Delete from linked list
3: Display linked list
4: count linked list
others: Exit()
Enter your option: 2
No. of items in linked list: 2
Display linked list:
Enter position for Delete element: 1
Deleted successfully
options:
1: Insert
2: Delete
3: Display
4: count
others: Exit()

Enter your option: 3
Display linked list:
3
No. of items in linked list: 1

options:
1: Insert into linked list
2: Delete from linked list
3: Display linked list
4: count linked list
others: Exit()

Enter your option: 5

3) find all the elements in the stack whose sum is equal to K (where K is given from user).

```
#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 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+=1;
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 in the stack dont add up
                                to the sum");
}
void push(int x)
{
if(top==99)
{
printf("\nstack is FULL !!!\n");
return;
}
top=top+1;
stack[top]=x;
```

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

output:

Enter the number of elements in the stack:4

Enter the next element 2

Enter next element 3

Enter next element 5

Enter next element 1

Enter the sum to be checked 15

The elements in the stack dont add up to the sum

4) write a program to print the elements in a queue
i) in reverse order.
ii) in alternate order.

ii)
```
#include<stdio.h>
#include<stdlib.h>
#define MAX 50
```

```
void insert();
void display();
int queue_array[MAX];
int rear = -1;
int rear = -1;
int main()
{
int choice;
while(1)
{
printf("1.Insert element to queue\n");
printf("2.Display alternate elements of queue\n");
printf("3.quit\n");
printf("Enter your choice:");
scanf("%d", &choice);
switch(choice)
{
case 1;
insert();
break;
case 2;
display();
break;
case 3;
exit(1);
default;
printf("wrong choice\n");
}
}
}
```

```
void insert ()
{
int item;
if (rear = = MAX - 1)
printf ("queue overflow n");
else
{
if (front = = -1)
front = 0 ;
printf ("Insert the Element in queue: ");
scanf ("%d", &item);
rear = rear + 1;
queue_array (rear) = item;
}
}
void display ( )
{
int i;
if (front = = -1)
printf ("queue is empty \n");
else
{
printf ("queue is empty \n");
else
{
print f ("queue is : \n");
for (i = front; i < rear; i = i+2)
printf ("%d", queue_array [i]);
printf ("\n");
}
}
```

output :

```
1 Insert element to queue
2 Display alternate elements of queue
3 Quit
Enter your choice : 1
Insert the element in queue : 2
1 Insert the element to queue
2 Display alternate elements of queue
3 quit
Enter your choice : 1
Insert the element in queue : 3
1 Insert element to queue
2 Display alternate elements of queue
3 Quit
Enter your choice : 2
queue is : 2
1 Insert element to queue
2 Display alternate elements of queue
3 Quit
Enter your choice : 3
```

ii)

```
#include <stdio.h>
#include <stdlib.h>
# define MAX 50
void insert();
void display();
int queue_array[MAX];
int rear = -1;
int front = -1;
int main()
{
int choice;
while(1)
```

```
{
printf("1. Insert Element to queue\n");
printf("2. Display Elements of queue in reverse
                                order\n");
printf("3. quit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
switch(choice)
{
case:1;
 insert();
 break;
 case:2;
 display();
 break;
 case:3;
 exit(1);
 default;
 printf("wrong choice\n");
}
}
}
void insert()
{
int item;
if(rear==MAX-1)
printf("Queue overflow\n");
 Else
{
int i
if(front==-1)
```

```
front=0;
printf("Insert the element in queue: ");
scanf("%d", &item);
rear = rear+1;
queue_array[rear] = item;
}
}
void display()
{
int i;
if (front == -1)
printf("Queue is empty\n");
else
{
printf("Queue is:\n");
for (i=rear; i>=front; i--)
print("%d", queue_array[i]);
printf("\n");
}
}
```

Output:

```
1 Insert Element to queue
2 Display Element of queue in reverse order
3 Quit
Enter your choice : 1
Inset the Element in queue: 2
1 Insert Element to queue
2 Display Element of queue in reverse order
3 Quit
Enter your choice: 1
Insert the Element in queue: 3
1 Insert Element to queue
2 Display Element of queue in reverse order
```

```
3 Quit
Enter your choice: 2
Queue is: 3 2
1 Insert Element to queue
2 Display Element of queue in reverse order
3 Quit
Enter your choice: 3
```

(5) (i) How array is different from the linked list

(ii) write a program to add the first Element of one list to another list for Example we have {1,2,3} in list 1 and {4,5,6} in list 2 we have to get {4,1,2,3} as output for list 1 and {5,6} for list 2.

(i) Differen 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 refre-nces to the previous and 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 = hed;
    while (ptr)
    {
      printf ("%d -> ", ptr->data);
      ptr = ptr->next;
    }
    printf ("NULL\n");
}
void push (struct Node** head, int data)
{
    struct Node* newNode = (struct Node*) malloc
        (size of (struct Node));
    newNode->data = data;
    newNode->next = *head;
      *head = newNode;
}
void MoveNode (struct Node** destRef, struct Node**
                    sourceRef)
{
  if (*sourceRef == NULL)
    return;
    Struct Node* newNode = *sourceRef;
    *sourceRef = (*sourceRef)->next;
    newNode->next = *destRef;
    *destRef = newNode;
}
  int main(void)
{
```

```
    int key[] = {1, 2, 3};
    int n = sizeof (keys) / sizeof (keys[0]);
    struct node * a = NULL;
    for (int i = n-1; i >= 0, i--)
        push (&a, key[i]);
  struct node* b = NULL;
    for (int i = 0; i < n; i++)
        push (&b, 2 * keys[i]);
    Move node (&a, &b);
   printf ("First list : ");
    print list (a);
    printf ("second list : ");
    print list (b);
    print
      return 0;
}
```

output :

First list : 6 -> 1 -> 2 -> 3 -> NULL

second list : 4 -> 2 -> NULL