DSA - Assignment-4

G. Sai Sushma,
AP19110010551,
CSE - G1.

1. 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 <stdlib.h>
 struct node
{
 int data;
 struct node *next;
 };
 display (strut node *head)
 {
  if (head == NULL)
 {
  printf ("NULL\n");
 }
  else
 {
  print f("%d\n", head -> data);
  display (head -> next);
 }
 }
del (struct node* before_del)
{
struct node * temp;
temp = before_del -> next;
before_del -> next = temp -> next;
 free (temp);
}
struct node * front (struct node* head, int value)
{
```

```
struct node *p;
p= malloc (size of (struct node));
p -> data = value;
p -> next = head;
return (p);
}
end (struct node * head, int value)
{
struct node *p, *q;
p = mallac (size of (struct node));
p -> data = value;
p -> next = NULL;
q = head;
while (q -> next != NULL)
{
q = q -> next;
}
q -> next = p;
}
after (struct node *a, int value)
{
if (a -> next != NULL)
{
struct node *p;
p = mallac (size of (struct node));
p -> data = value;
p -> next = a -> next;
a -> next = p;
}
else
{
printf ("USE END FUNCTION TO INSERT AT THE END\n");
}
}
```

```
int main ( )
{
 struct node *prev, *head, * p;
 int a, i;
 printf ("NUMBER OF ELEMENTS");
 scanf (" %d ", &a);
 head = NULL;
 for (i=0; i<a; i++)
 {
 p=malloc (sizeof (struct node));
 scanf (" %d", &p -> data);
  p -> next = NULL;
  if (head = NULL)
   head = p;
  else
  prev -> next = p;
  prev = p;
  }
  head = front (head, 10);
  end (head, 20);
  after (head -> next -> next, 30);
   del (head -> next);
   del ( head -> next -> next);
   display (head);
   return 0;
  }
```

out put:

```
NUMBER OF ELEMENTS 5
1
2
3
4
5
```

10
1
30
4
5
20
NULL

2) Construct a new linked list by merging alternate nodes of two lists for example in list by meaning 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 data;
 struct Node* next;
};
void push (struct Node ** head_ref, int new_data)
{
struct Node *new_node =(struct Node*) mallac(size of (structNode));
{while (temp! = NULL)}
new_node -> data = new_data;
new_node -> next = (*head_ref);
(*head_ref) = new_node;
}
void print list (struct Node *head)
{
struct Node *temp = head;
while (temp! = NULL)
{
printf ("%d", temp -> data);
temp= temp -> next;
}
print f ("\n");
}
void merge (struct Node *p, struct Node **2)
```

```
{
struct Node* P_curr = P, *q_curr = *q;
struct Node* P_next, * q_next;
while (P_curr != NULL && q_curr != NULL)
P_next = P_curr->next;
q_next = q_curr->next;
q_curr->next = P_next;
P_curr->next = q_curr;
P_curr = P_next;
q_curr = q_next;
}
*q = q_curr;
}
int main()
{
struct Node *P = NULL, *q = NULL;
push(&P, 0);
push(&P, 3);
push(&P, 1);
~~push(&P~~
Printf("1st LINKED LIST\n");
Print list(P);
push(&q, 2);
push(&q, 8);
push(&q, 0);
printf("2nd LINKED LIST\n");
print list(q);
merge(P, &q);
Printf("CHANGED LINKED LIST\n");
print list(P);
return 0;
}.
```

out put:

1st LINKED LIST

1 3 0

2nd LINKED LIST

0 8 2

~~2nd~~ LINKED LIST
(UNSORTED

10 3 8 0 2

3. Find all the dements in the stack whose sum is equal to K.

```
#include <stdio.h>
#include <stdlib.h>
#include <limit.h>
#define max 1000
typedef struct STACK{
  int ar[max];
  int top;
} stack;

void push (stack *s, int data){
   if (s->top) >= max-1) {
      exit(0);
    }
     s->top++;
     s->ar[s->top] = data;
}
  int pop (stack *s) {
     if (s->top < 0) return INT_MIN;
     int temp = s->ar[s->top];
        s->top--;
        return temp;
```

```
}
void display (stack s) {
  int i;
  for (i=s.top; i>-1; i--){
    printf ("%d", sar[i]);
  }
  printf ("\n");
}
void sumk (stack s1, stack v, int k){
 if (k==0) {
  display (v);
  return;
 }
 if (s1.top == -1) return;
 int temp = pop (&s1);
  sumk (s1, v, k);
   stack v1 = v;
   push (&v1, temp);
   sumk (s1, v1, k-temp);
}
int main (int argc, char const *argv[]) {
stack arr, v;
arr.top = -1;
 v.top = -1;
 int expected, n, num;
 printf ("enter the number of element of element you want in
          the stack \n")
  scanf ("%d", &n);
  while (n--){
  printf ("numbo\n");
  scanf ("%d", &num);
  push (&arr, num);
}
```

```
printf ("enter expected value\n");
scanf ("%d", &expected);
n = arr.top+1
sumk (arr, v, expected);
return 0;
}
```

output:

Enter the number of element you want in the stack

10
number
1
number
2
number
3
number
7
number
8
number
9
number
7
number
5
number
5
number
6

Enter expected values

15
7 8
1 2 3 9
1 2 3 8 1
2 3 9 1
1 2 7 5
3 7 5

2 8 5
1 9 5
2 7 1 5
1 8 1 5
9 1 5
7 8

1 2 3 1 8
2 5 8
1 1 5 8
2 7 6
1 8 6
9 6
1 2 1 6

8 8 6
1 3 5 6
1 2 1 5 6
3 1 5 6
1 8 6
1 8 6

4. write a program to print the elements in a queue in reverse order and alternate order.

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node{
  int data;
  struct Node *next;
} node;

typedef struct Queue{
  node *front, *rear;
} queue;

node *temp = (node*) malloc(sizeof(node));
temp->data = k;
temp-> next = NULL;
return temp;
}

queue createQueue()
{
 queue q;
 q.front = q.rear = NULL;
 return q;
}
void enQueue(queue *q, int k)
{
 node *temp = newNode(k);
 if (q->rear == NULL){
   q->front = q->rear = temp;
  return;
}
```

```
    q->rear->next = temp;
    q->rear = temp;
}

void displayAll(queue q){
    while(q.front != NULL){
    printf("%d ->->", q.front->data);
    if(q.front->next != NULL) q.front = q.front->next->next;
     else break;
    }
    printf("NULL\n");
}
 void displayRev(queue q){
    if(q.front == NULL){
     printf("NULL");
      return;
    }
    int temp = q.front->data;
     q.front = q.front->next;
     displayRev(q);
     printf("<-%d", temp);
 }
 int main()
 { queue q = createQueue();
    int n, num;
    printf("enter the number of element you want in the queue\n");
    scanf("%d", &n);
    while(n--){
      printf("number\n");
      scanf("%d", &num);
      enQueue(&q, num);
    }
     displayRev(q);
     printf("\n");
     displayAll(q);
     return 0;
    }
```

output:

Enter the number & element you want in queue

6

number

1

number

2

number

3

number

4

number

5

number

6

NULL ← 6 ← 5 ← 4 ← 3 ← 2 ← 1

1 → → 3 → → 5 → → NULL

5) i) difference between array and linked list.

arrays	linked list
1. fixed size; Re sizing is expensive	1. Dynamic size
2. Insertions and Delections are inefficients. elements are usually shuffled.	2. Insertion and deletions are effecient: no shuffling
3. Random access i.e efficient indexing	3. NO random access
4. No memory waste if the array is full or almost full; otherwise may result in much waste memory.	4. Not suitable for operations requiring accessing elements by index such as sorting. since memory is allocated dynamically no memory wastage

5. Sequential access is fast	5. sequential access is slow.

3) ii) write a program to add the first element of 1 list to another list.

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
 int data;
 struct Node *next;
};
void push (struct Node ** head_ref, int new_data)
{
struct Node * new_node = (struct Node*) malloc (size of (struct Node));
new_node -> data = new_data;
new_node -> next = (*head_ref);
(*head_ref ) = new_node;
}
void pointlist (struct Node * head)
{
 struct Node* temp = head;
 while (temp ! = NULL)
 {
 printf ("%d", temp -> data);
 temp = temp -> next;
 }
 printf ("\n");
 }
void merge (struct Node *p, struct Node **q)
{
struct Node * p_curr = p, *q_curr = *q;
 struct Node * p_next, *q_next;
```

```
while (p_curr != NULL && q_curr != NULL)
p_next = p_curr -> next;
q_next = q_curr -> next;
q_curr -> next = p_next;
p_curr -> next = q_next;
p_curr = p_next;
q_curr = q_next;
}
*q = q_curr;
}
int main()
{
struct Node *p = NULL, *q = NULL;
push(&p, 0);
push(&p, 3);
push(&p, 1);
printf("1st LINKED LIST\n");
printlist(p);
push(&q, 1);
push(&q, 7);
push(&q, 2);
push(&q, 8);
push(&q, 0);
printf("2ND LINKED LIST\n");
printlist(q);
merge(p, &q);
printf("CHANGED 1st LINKED LIST\n");
printlist(p);
```

```
printf ("CHANGED 2ND LINKED LIST\n");
print list (q);
```

Output:

1st LINKED LIST

2 6 3

2ND LINKED LIST

3 4 5 1 7

CHANGED 1st LINKED LIST

7 3 6 4 3 5

CHANGED 2ND LINKED LIST

1 2.

* THE END *