NAME :- SUYASH DAYAL

ROLL :- AP19110010530

CLASS :- CSE:-H

Date ___/___/_____

Assignment : 4

Q1. Write a program to insert and delete an element at the n^{th} and k^{th} pointer in a linked list where n and k are taken from the users.

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

struct node
{
int element;
struct node * next;
};
struct node * head;

void insert (int element, int n)
{
node * temp = new node ();
temp -> element = element;
temp -> next = NULL;
if (n==1)
{
```

```
temp → next = head;
head = temp;
}

void delete (int a)
{
struct node * temp = head;
if (a==1)
{
head = temp → next;
free (temp);
}
node * temp = head;
for (int i = 0; i < n-2; i++)
{
temp = temp → next;
}
temp → next = temp → next;
temp → next = temp;
}

void print ( );
{
for (int i = 0; i < a-2; i++)
{
```

```
temp = temp → next ;
free (temp);
}
}

void main()
{
int n, e, K;
head = NULL;
printf (" Enter the element and its position");
scanf ("%d-%d", &e, &n);
insert (e, n);
printe (" Enter the position for deletion");
scanf ("%d", &K);
delete (K);
print (e);
}
```

OUTPUT :

Enter the element and its position 53 4
Enter the position for deletion 4

Date __/__/____

4.

Q2. Construct a new linked list by merging alternate 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 element;
struct node * next;
};

void display (struct node * head)
{
struct node * ptr = head;
while (ptr)
{
printf ("%d", ptr -> element);
ptr = ptr -> next;
}
printf("NULL");
}
```

```
void insert (struct node * head, int element)
{
struct node * newnode = (struct node *)
                        malloc(sizeof(struct node));
newnode -> element = element;
newnode -> next = * head;
* head = newnode;
}

struct node * altmerge (struct node * a,
                        struct node * b)
{
struct node alt;
struct node * tale = & alt;
alt.next = NULL;
while (1)
{
if (a == NULL)
{
tale -> next = b;
break;
}
else if (b == NULL)
{
tale -> next = a;
```

```
break;
}
else
{
tale → next = a;
tale = a;
a = a → next;
tale → next = b;
tale = b;
b = b → next;
}
}
return alt.next;
}

void main()
{
int data[] = {1,2,3,4,5,6,7};
int n;
n = sizeof(data)/sizeof(data[0]);
struct node *a = NULL;
struct node *b = NULL;
for (int i=n-1; i>=0; i=i-2)
{
insert(&a, data[i]);
```

```
for(int i = n-2; i >=0; i = i-2)
{
insert (&b, data[i]);
}

printf (" First list is : ");
display (a)
printf (" second list is: ");
display (b);
struct node * head = altmerge (a,b);
printf (" After merging : ");
display (head);
}
```

OUTPUT :

First list is 1 2 3

Second list is 4 5 6

After merging 1 4 2 5 3 6

Q3. Find all the elements in the stack whose sum is equal to K. (where K is given from the user).

```
#include <stdio.h>

int x, top = -1;
int stack[100];

void push (int x)
{
if(top == 99)
{
printf (" Stack is FULL OVERFLOW !! ");
}
top = top+1;
stack[top] = x;
}

char pop()
{
if (stack[top] == -1)
{
printf (" Stack is empty UNDERFLOW !! ");
}
```

```
x= stack [top];
top= top-1;
return x;
}

void main()
{
int i,n,a,p,s,r, sum=0, count=1;
printf (" Enter number of elements:");
scanf ("%d", &n);
for (i=0; i<n; i++)
{
printf (" Enter element:");
scanf ("%d", &a);
push (a);
}
printf (" Enter the sum to be checked");
scanf ("%d", &s);
for (i=0; i<n; i++)
{
p= pop()
sum+ =p
count+ =1
if (sum==s)
{
```

```
for(int j=0; j<count, j++)
printf ("%d", stack [j]);
r=1;
break;
}
push (p);
}
else
{
printf (" The elements in the stack
        does not add up to this sum");
}
```

OUTPUT:

```
Enter number of elements: 3

Enter element: 1
Enter element: 5
Enter element: 2

Enter the sum to be checked 6
1  5
```

Date ___/___/_____

Q4. Write a program to print the elements in a queue :

(i) In reverse order

(ii) In alternate order.

Ans :

```
# include <stdio.h>
# define SIZE 10
int queue [SIZE], f=-1, r=-1;

void enqueue (int value)
{
if ((f==0 && r= SIZE-1) || f==r+1)
{
printf (" OVERFLOW");
}
else
{
if(f==-1)
 f=0;
r=(r+1)% SIZE
queue[r] = value;
printf(" Insertion successful");
}
}
```

```
void dequeue ()
{
if (f==-1)
printf ("UNDERFLOW");
else
{
printf ("Deleted element", queue[f]);
f= (f+1)%SIZE
if (f==r)
f=r=-1
}
}

void main()
{
int value, choice;
while(1)
{
printf ("1. Insertion 2. Deletion 3. Print
        reverse 4. Print alternate 5. Exit");
scanf ("%d", &choice);
switch (choice)
{
case 1: printf ("Enter the value to be inserted");
```

Date ___/___/_____

```
scanf("%d", &value);
enqueue(value);
break;

case 2: dequeue();
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=i+2)
{
if(queue[i]==0)
continue;
printf("%d", queue[i]);
}
break;
```

```
case 5: exit(0);
default : printf("Invalid input");
}
}
}
```

OUTPUT:

```
1. Insertion  2. Deletion  3. print reverse
4. print alternate  5. Exit.
1
Enter the value to be inserted. 3
1
Enter the value to be inserted 5
3
5  3
4
3
```

Date ___/___/_____

Q5. (i) How array is different from linked list

ARRAY	LINKED LIST
(i) Array's size is specified during its declaration	(i) The size of a linked list can be grown or shrinked during execution.
(ii) It is a consistent set of a fixed number of data items.	(ii) It is an ordered set comprising a variable number of data items.
(iii) It is stored consecutively.	(iii) It is stored randomly.
(iv) It requires less memory.	(iv) It requires more memory.
(v) It can be directly or randomly accessed using array index.	(v) It can be sequentially accessed.

(ii) Write a program to add the first element of one list to another list for example we have {1,2,3} is list1 and {4,5,6} is list 2 then we have to get {4,1,2,3} as output for list 1 and {5,6} for list 2

Ans:

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

int len (int a[])
{
int i=0, p=0;
while (1)
{
if (a[i])
{
p++;
i++;
}
else
{
break;
}
}
return p;
}
```

```
void changelist ( int a[], int b[])
{
for ( int i = len(a)-1 ; i>=0 ; i--)
{
a [i+1] = a[i] ;
}
a[0] = b[0] ;
printf (" The elements of first array : ");
for (int i=0 ; i < len(a) ; i++)
{
printf ("%d", a[i]);
}
for (int i=0 ; i < len(b) ; i++)
{
b[i] = b[i+1] ;
}
printf (" The elements of second array : ");
for (int i=0 ; i < len(b) ; i++)
{
printf ("%d", b[i]);
}
}
```

```
Void main()
{
int a[10] = {1,2,3} , b[10]= {4,5,6};
changelist (a,b);
}
```

OUTPUT:-

The elements of first array : 4 1 2 3

The elements of second array: 5 6

—x—