R. Ramya Manasa
AP19110010472
CSE-F

Assignment-4

1. Write a C program to insert and delete an element at the nth and Kth position in a linked list where n and K is taken from the User.

Code:-

```
#include<stdio.h>
#include <stdlib.h>
Void Create();
Void display();
Void insert_pos();
Void delete_pos();
Struct node
{
        int info;
        Structnode *next;
};
Structnode *start = NULL;
int main()
{
      int choice;
      while(1){
            printf(" Menu");
            printf("\n1. Create\n");
            printf("\n 2. Display \n");
            printf("3.Insert a node at Specified position \n");
            printf ("4.Delete node \n");
            printf ("5.Exist \n");
```

```
printf ("Enter your choice : \t");
Scanf ("%d", &choice);
switch(choice)
{
    case 1:
            create();
            break;
    case 2:
            display();
            break;
    case 3:
            insert_pos();
            break;
    case 4:
            delete_pos();
            break;
    case 5:
            exist(0);
            break;
    defalt:
            printf("\n wrong choice : \n");
            break;
    }
}
return 0;
}
```

```
Void Create()
{
    Struct node *temp, *ptr;
    temp = (struct node*) malloc(size of (struct node));
    if(temp==NULL)
    {
        printf("\n Out of Memory space: \n");
        exit(0);
    }
    printf("\nEnter the data value for the node: \t");
    Scanf("%d", &temp->info);
    temp->next = Null;
    if(start == NULL)
    {
        start = temp;
    }
    else{
        ptr = start;
        while(ptr->next != NULL)
        {
            ptr = ptr->next;
        }
        ptr->next = temp;
    }
}
Void display()
{
    Struct node *ptr;
    if(start == NULL)
```

```
        {
            printf("\n List is empty:\n");
            return;
        }
        else
        {
            ptr=start;
            printf("\n The List elements are: \n");
            While (ptr !=NULL)
            {
                printf("%d\t", ptr-> info);
                ptr= ptr->next;
            }
        }
}
Void insert_pos()
{
    Struct node *ptr,*temp;
    int i, pos;
    temp = (structnode*)malloc(size of (struct node));
    if (temp==NULL)
    {
        printf("\n Out Of Memory Space: \n");
        return;
    }
    printf ("\n Enter the position for new node to be inserted:\t");
    Scanf("%d", &pos);
    printf ("\n Enter the data value of the node: \t");
    Scanf ("%d", &temp -> info);
```

```
        temp->next = NULL;
        if(pos == 0)
        {
            temp->next = start;
            start = temp;
        }
        else
        {
            for(i=0; ptr=start; i<pos-1; i++)
            {
                ptr = ptr->next;
                if (ptr == NULL)
                {
                    printf("\n position not found: \n");
                    return;
                }
            }
            temp -> next = ptr -> next;
            Ptr -> next = temp;
        }
}
Void delete_pos( )
{
        int i, pos;
        Struct node *temp, *ptr;
        if(start == NULL)
        {
            Ptr = start;
```

```
        start = start -> next;
        printf("\nThe deleted element is:%d\t", ptr->info);
        free(ptr);
    }
    else
    {
        ptr = start;
        for(i=0;i<pos;i++)
        {
            temp= ptr;
            ptr= ptr->next;
            if(ptr==NULL)
            {
                printf("\n position notFound: \n");
                return;
            }
        }
        temp-> next = ptr->next;
        printf("\n The deleted element is :%d\t", ptr->info);
        free (ptr);
    }
}
}
```

Output:

Menu

1. Create
2. Display
3. Insert a node at Specified position.
4. Delete node
5. Exist.

Enter your choice: 1
Enter the data value for the node: 10

Menu

1. Create
2. Display
3. Insert a node at Specified position
4. Delete node
5. Exist

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. Delet node
5. Exist

Enter your choice:3
Enter the position for the new node to be inserted: 2
Enter the data value of the node:33

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.

Menu

1. create
2. Display
3. Insert at Specified posified position
4. Delete
5. Exist

Enter your choice : 5

2. Construct a new linked list by merging alternative nodes of two lists.

Code:

```
#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* newNode = (Struct Node*) malloc(size of-(Struct Node));
    new Node -> data =data;
    newNode -> next = * head;
    * head = new Node;
}
```

```
Struct Node* shuffleMerge(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(keys)/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 → NULL.

3. Find all the elements in the Stack whose Sum is equal to - K.

```
#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+=t;
count++;
```

```
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 is not equal to sum");
}
void push(int x)
{
if(top==99)
{
  printf("\nstack is full!!\n");
return;
}
top=top+1;
stak[top]=x;
}
char pop()
{
if(stack[top]==-1)
{
```

```
Printf("\n Stack is Empty !!! \n");
return 0;
}
x = stack[top];
top = top - 1;
    return x;
}
```

Input - Output:

```
Enter number of elements in stack 3
Enter element 2
Enter element 3
Enter elemen 4
Enter the Sum to be checked 20
The elements in the stack donot equal to Sum.
```

4) Write a program to print the elements in a queue
i. in reverse order
ii. in 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 \n2.Deletion \n3.print reverse \n
            4. print Alternate\n 5.Exit");
        printf("\n Enter your choice: ");
        Scanf("%d", &choice);
        Switch(choice){
            case 1: printf("Enter value to insert: ");
                Scanf("%d",&value);
                insert(value);
                break;
            Case 2: delete();
                breat;
            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 : ");
    for(int i=0; i<SIZE; i+=2)
    if(queue[i]==0)
     continue;
     printf("%d ", queue[i]);
     break;
 Case 5:
        exit(0);
 default :
     printf("In wrong selection!!! Try again!!!");
  }
   }
}
Void insert(int value) {
    if ((f==0 && r==SIZE-1) || f==r+1)
      printf("In 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 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;
        }
 }
```

Code:

MENU

1. Insertion
2. Deletion
3. print Reverse
4. print Alternate
5. Exit

Enter your Choice : 1

Enter the value to be insert : 5

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 : 10

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 : 15

Insertion Success!!!

Menu

1. Insertion
2. Deletion
3. print Reverse
4. print Alternate
5. Exit.

Enter your choice: 3
The Reversed queue is: 15 10 5

MENU

1. Insertion
2. Deletion
3. print Reverse
4. print Alternate
5. Exit

Enter your choice: 4
Alternate elements of the queue are: 5 15

MENU

1. Insertion
2. Deletion
3. print Reverse
4. print Alternate
5. Exit

Enter your choice: 5

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

The major difference between Array and Linked list regards to their structure. Arrays are index based data structures 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 references to the previous and next element.

ii, Write a program to add the first element of one list to another list.

Code:

```
#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* newNode = (structNode *)malloc(sizeof(struct Node));
        newNode → data = data;
        new Node → 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 keys[] = {4, 3, 8};
    int n = sizeof(keys) / sizeof(keys[0]);
    struct Node* a = NULL;
    for (int i=n-1; i>=0; i--)
        push(&a, keys[i]);
    struct Node* b = NULL;
    for (int i=0; i<n; i++)
        push(&b, 3*keys[i]);
    MoveNode(&a, &b);
    printf("First List: ");
    printList(a);
    printf("Second List: ");
    printList(b);
    return 0;
}
```

Input-Output:

First List: 24 → 4 → 3 → 8 → NULL

Second List: 9 → 12 → NULL