N. Lakshmi Manasa ①
CSE-F
AP19110010382

DSA ASSIGNMENT-4

1. Write a program to insert and delete element at nth and kth position in a linked list where n and k is taken from user.

Program:-

```
# include<stdio.h>
# include<stdlib.h>
struct node
{
    int n;
    struct node *next;
};
struct node *curr, *temp;
struct node *create(struct node*);
void inspos(struct node*);
void delpos(struct node*);
void main(void)
{
struct node *s;
int ch;
s= NULL;
do
{
printf("\n 1. Create\n");
printf("2. Inspos\n");
printf("3. delpos\n");
printf("4. Exit\n");
printf("Enter the choice");
scanf("%d", &ch);
    switch(ch)
    {
    case 1: s=create(s);
            break;
    case 2: inspos(s);
            break;
    case 3: delpos(s);
            break;
} which(ch!=4)
}
struct node* create(struct node *x)
            if(x=NULL)
            {
              x=(struct node*) malloc(size of (struct node));
              printf(" Enter the number");
              scanf(" %d", &x->n);
              x->next=NULL;
              return x;
            }
            else
             {
              printf(" The node already created");
              return x;
             }
 }
```

```
void inpos(struct node *x)
{
        int pos, c=1;
        curr=x;
        printf("Enter the pos to be inserted:");
        scanf("%d", &pos);
           while(curr->next != NULL)
           {
           c++;
              if(c==pos)
              {
               temp= (struct node*)malloc(size of(struct node));
               printf("Enter the number:");
               scanf("%d", &temp->n);
               temp->next = curr->next;
               curr->next=temp;
               break;
               }
           }
}
void delpos (struct node *x)
{
        int pos, c=1;
        curr=x;
        printf("Enter the pos to be deleted:");
        scanf("%d", &pos);
        while(curr->next! = NULL)
        {
        c++;
                if(c==pos)
                {
                    temp=curr->next;
                    curr->next=curr->next->next;
                    free(temp);
                    }
                curr=curr->next;
}
```

Output:-

```
1. Create
2. Inspos
3. delpos
4. Exit
Enter the choice 1
Enter the number 4'5

1. Create
2. Inspos
3. delpos
4. Exit

Enter the choice 2
Enter the pos to be inserted 2
Enter the number 6
1. Create
2. Inpos
3. delpos
4. Exit
```

2. Construct a new linked list by merging alternate nodes of two lists for example in list1 we have {1,2,3} and in list2 we have {4,5,6} in the new list we should have {1,4,2,5,3,6}.

Program:-

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

struct node
{
   int data;
   struct Node* next;
};
void MoveNode(struct Node** destRef, struct Node** sourceRef);
struct Node* SortedMerge(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;
            }
            if (a -> data < = b -> data)
               MoveNode (&(tail -> next), &a);
            else
               MoveNode (&(tail -> next), &b);

            tail = tail -> next;

     }
     return (dummy.next);
}
void MoveNode(struct Node** desRef, struct Node** sourceRef)
{
   struct Node* newNode = *sourceRef;
   assert(new Node ! = NULL);
   * Source Ref = newNode -> next;
   newNode -> next = * desRef;
   * desRef = new Node;
}
void push(struct Node** head-ref, int new-data)
{
    struct Node* new-node = (struct Node*) malloc(sizeof (struct Node));
    new-~~Node~~ -> data = new_data;
    new_node -> data = (* head-ref);
    (* head-ref) = new_node;
```

```
}
Void printList (struct Node *node)
{
    while (node != NULL)
    {
      printf(" %d", node->data);
      node = node -> next;
    }
}
int main()
{
    struct Node* res = NULL;
    struct Node* a = NULL;
    struct Node* b = NULL;
  /* Created lists, a: 1->2->3, b: 4->5->6*/
      push(&a,1);
      push(&b,4);
      push(&a,2);
      push(&b,5);
      push(&a,3);
      push(&b,6);
      res = sortedMerge(a,b);
      printf (" Merged Linked List is: \n");
      printList(res);
      return0;
}
```

Output:-

```
Merged linked list is
1 4 2 5 36.
```

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

Sol Program:-

```
#include<stdio.h>
int top=-1;
int j;
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);
    }
    t= pop();
    Sum+ =t;
    Count+=1;
    if (sum==k)
     {
        for(j=0; j<count; j++)
        printf("%d", stack[j]);
        f=1;
        break;
     }
     push(t);
   }
  if (f!=1)
     prinf (" The elements in the stack donot addup to the 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 is EMPTY!!!\n"),
 return 0,
}
x= stack[top];
top = top-1;
 return x;
```

Output:-

```
Enter the number of elements in the stack 3
Enter next element 2
Enter next element 5
Enter next element 3
Enter the sum to be checked 11
The elements in the stack donot add up to the sum.
```

4. Write a program to print the elements in a queue.

i. in reverse order

ii. in alternate order.

Program:-

```
# 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(" 1. Insertion\n 2.Deletion\n 3. Print Reverse\n
                                          4. Print Alternate\n3. Exit ");
               printf ("\n Enter your choice: ");
               scanf ("%d", &choice);
               switch(choice)
               {
     case1: printf(" Enter the value to be insert:  ");
            scanf (" %d", &value);
            insert (value);
            break;
     case2: delete();
            break;
     case3: printf (" The Reversed Queue is: ");
            for(int i=SIZE;i>=0;i--)
            {
                if(queue[i]=0)
                continue;
                printf(" %d", queue[i]);
            }
            break;
```

5.

```
case 4 : printf(" Alternate elements of queue are: ");
        for (i=0; i< SIZE; i+=2)
        {
            if (queue[i] =0)
            continue;
            printf ("%d", queue[i]);
        }

        break;
    case 5: exit(0);
    default: printf ("\n Wrong choice!! Try again!! ");
      }
}}
void insert(int value)
{
   if((f==0&&r== SIZE-1) || f= r+1)
        printf(" \n Queue is full!! Insertion is not possible!! ");
   else {
       if (f==-1)
f= 0;
      r= (r+1) % SIZE;
      queue[r]=value;
      printf (" 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
}}.
```

Output:-

```
1. Insertion
2. Deletion
3. Print Reverse
4. Print Alternate
Enter your choice:1
Enter the value to be insert:10
Insertion Success!!

1. Insertion
2. Deletion
3. Print Reverse
4. Print Alternate
Enter your choice: 1
Enter the value to be insert: 20
Insertion Success!!
```

```
1. Insertion
2. Deletion
3. Print Reverse
4. Print Alternate
Enter the value to be insert: 30
Insertion success!!
1. Insertion
2. Deletion
3 Print Reverse
4. Print Alternate
Enter your choice : 3
The reversed queue is 30 20 10.
1. Insertion
2. Deletion
3. Print Reverse
4. Print Alternate
Enter your choice: 4
Alternate elements of the queue are : 10 30.
```

5.

i. How array is different from the linked list

1) An array consumes contiguous memory locations allocated at compile time, i.e., at the declaration of array, whereas for linked list, memory is assigned as and when data is added to it, which means at runtime.

2) Another difference between array and linked list is based on 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 references to the previous and next element.

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.

Program:-

```
# 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 printList(struct Node *head)
{
    Struct Node * temp=head;
    while(temp! =NULL)
    {
        printf("%d", temp->data);
        temp = temp->next;
    }
    printf("\n");
}
void merge(struct Node *x, struct Node *y)
{
    struct Node * x_curr = x, *y_curr = *y;
    Struct Node *x_next, *y_next;
    while(x_curr! =NULL&& y_curr != NULL)
    {
            x_next = x_curr->next;
            y_next = y_curr->next;

            y_curr->next = x_next;
            x_curr->next = y_curr;

            x_curr = x_next;
            y_curr = y_next;
        }
        *y = y_curr;
}
int main()
{
    struct Node *x = NULL, *y = NULL;
    push(&x, 3);
    push(&x, 2);
    push(&x, 1);
    printf("First linked list:\n");
    printlist(x);
    push(&y, 8);
    push(&y, 7);
    push(&y, 6);
    push(&y, 5);
    push(&y, 4);
    printf("Second linked list:\n");
    printlist(y);
```

```
merge(&x, &y);
printf("Modified First linked list:\n");
printlist(x);
printf("Modified second linked list:\n");
print list(y);
getchar();
return0;
}
```

Output:-

```
First linked list:
1 2 3
Second linked list:
4 5 6 7 8
Modified First linked list:
1 4 2 5 3 6
Modified Second linked list:
7 8.
```

[illegible]