DSA ASSIGNMENT

G. Mitra Datta
AP19110010552
CSE-H

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 are taken from user.

Sol:-

```
#include<stdio.h>
#include<malloc.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node * next;
};
struct node *head = NULL;
void insert(int n)
{
    struct node * newnode;
    int i;
    newnode = (struct node *) malloc (sizeof (struct node*));
    printf ("Enter Data:");
    scanf (" %d", &newnode -> data);
    if (n==1)
    {
        newnode -> next = head
        head = newnode;
    }
    else
    {
        struct node *temp = head;
        for(i=1; i<=n; i++)
            temp = temp -> next;
        newnode -> next = temp -> next;
        temp -> next = newnode;
    }
}
```

```
void delete(int k)
{
    struct node *temp1=head;
    if(k==1)
    {
        head=temp1->next;
        free(temp1);    // Optional, deleting unwanted memory
    }
    else
    {
        int i;
        for(i=1; i<k-1; i++)
            temp1=temp1->next;
        struct node *temp2=temp1->next;
        temp1->next=temp2->next;
        free(temp2);
    }
}
void display()
{
    struct node* newnode;
    newnode=head;
    printf("Linked list is:\n");
    while(newnode!=NULL)
    {
        printf("\n%d",newnode->data);
        newnode=newnode->next;
    }
}
```

```
void main()
{
    int n,k,choice;
    while(1)
    {   printf("1. Insert In 2. Delete In. 3. Display 4. Exit");
        printf("Enter choice:");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: printf("Enter position to insert:");
                    scanf("%d",&n);
                    insert(n);
                    break;
            case 2: printf("Enter position to delete:");
                    scanf("%d",&k);
                    delete(k);
                    break;
            case 3: display();
                    break;
            case 4: exit(0);
            default: printf("Wrong Input");
        }
    }
}
```

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

Sol:-

```
#include<stdio.h>
#include<stdlib.h>
#include<malloc.h>
struct node
{
    int data;
    struct node *next;
};
void insert_at_begin(struct node **head, int data)
{
    struct node *newnode = (struct node *)malloc(sizeof
                                                 (struct node));
    newnode->data = data;
    newnode->next = *head;
    *head = newnode;
}
void display(struct node *head)
{
    struct node *temp = head;
    while(temp)
    {
        printf("%d->", temp->data);
        temp = temp->next;
    }
}
```

```
void merge(struct node **a, struct node **b)
{
    struct node temp;
    struct node *tail=&temp;
    temp.next=NULL;
    while(1)
    {
        if(*a==NULL)
        {
            tail->next=NULL;
            break;
        }
        else if(*b==NULL)
        {
            tail->next=*a;
            [illegible]
            break;
        }
        else
        {
            tail->next=*a;
            tail=*a;
            *a=(*a)->next;
            tail->next=*b;
            tail=*b;
            *b=(*b)->next;

        }

    }
    *a=temp.next;
}
```

```
void main()
{
    int i,x,y,m,n;
    struct node *list1=NULL, *list2=NULL;
    printf("Enter number of elements to insert
            for list1:");
    scanf("%d",&x);
    for(i=1; i<=x; i++)
    {
        printf("Enter element to insert:");
        scanf("%d",&m);
        insert_at_begin(&list1,m);
    }
    printf("First list is:\n");
    display(list1);
    printf("Enter number of elements to insert
            for list2:");
    scanf("%d",&y);
    for(i=1; i<=y; i++)
    {
        printf("Enter element to insert:");
        scanf("%d",&n);
        insert_at_begin(&list2,n);
    }
    printf("Second list is:\n");
    display(list2);
    merge(&list1,&list2);
    printf("After Merging:\n");
    printf("Merged list is:\n"); // New list
    display(list1);
}
```

4) Write a C program to print the elements in a queue in

(i) reverse order (ii) in alternate order

Sol:-

```
#include<stdio.h>
#include<stdlib.h>
#define size 20
int queue[20], front=-1, rear=-1;
void enqueue(int n)
{
    if(rear==size-1)
        printf("Queue is full");
    else
    {
        if(front==-1)
            front=0;
        rear++;
        queue[rear]=n;
    }
}
void dequeue()
{
    if(front==rear)
        printf("Queue is empty");
    else
    {
        printf("\n deleted: %d", queue[front]);
        front++;
        if(front==rear)
            front=rear=-1;
    }
}
```

```
void displayreverse()
{
    if(rear==-1)
        printf("Queue is Empty");
    else
    {
        int i;
        printf("\n Queue is:: \n");
        for(i=rear; i>=front; i--)
            printf("%d ", queue[i]);
    }
}
void displayalternate()
{
    if(rear==-1)
        printf("Queue is Empty");
    else
    {
        int i;
        printf("Queue alternate elements are\n");
        for(i=front; i<=rear; i+=2)
            printf("%d ", queue[i]);
    }
}
```

```
void main()
{
    int n,choice;
    while(1)
    {
        printf(" 1.Enqueque \n 2.Dequeue.\n 3. Displayreverse \n
            4. Displayalteriate \n");

        printf(" Enter your choice:");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: printf(" Enter element to insert:");
                    scanf("%d",&n);
                    enqueue(n);
                    break;
            case 2: dequeue();
                    break;
            case 3: displayreverse();
                    break;
            case 4: displayalternate();
                    break;
            case 5: exit(0);
            default: printf("Wrong input");
        }
    }
}
```

5.)

i) Differences between array and linked list

Array	Linked List
① An array is a collection of elements stored in adjacent memory locations.	① Linked list is an order collection of nodes which have two parts data, next. Nodes are elements connected by pointers.
② Random access is possible in array	② Random access is not possible in linked list. Elements can be accessed orderly.
③ Array uses static memory allocation (fixed memory)	③ Linked list uses dynamic memory allocation. (memory allocated at execution)
④ Size of array must be specified during initialisation.	④ Size of linked list is adjusted according to insertion, deletion.

(ii) Write a C program to add the first element of one linked list to another linked list.

Sol:-

```
#include<stdio.h>
#include<stdlib.h>
#include<malloc.h>
struct node
{
    int data;
    struct node *next;
};
void insert_at_begin(struct node **head,int data)
{
    struct node *newnode=(struct node*)malloc(
                                sizeof(struct node));
    newnode->data=data;
    newnode->next=*head;
    *head=newnode;
}
void movenode(struct node **destination,struct node**source)
{
    if(*source==NULL)
        return;
    struct node *newnode=*source;
    *source=(*source)->next;
    newnode->next=*destination;
    *destination=newnode;
}
```

```
void display(stvuct node *head)
{
    struct node *temp=head;
    while(temp)
    {
        printf(" %d -> ", temp->data);
        temp= temp->next;
    }
}
void main()
{
    int n1,n2, x, y, i;
    printf(" Enter number of nodes in list1:");
    scanf(" %d ",&n1);
    struct node * list1=NULL;
    for(i=1; i<=n1; i++)
    {
        printf(" Enter element to insert into list1:");
        sconf("%d", &x);
        Insert_at_begin( &list1, x);
    }
    printf ("Enter number of nodes in list2:");
    scanf ("%d", &n2);
    for(i=1; i<=n2; i++)
    {
        printf(" Enter element to insert into list2:");
        sconf(" %d", &y);
        insert_at_begin( &list2, y);
    }
    movenode (&list1, &list2);
    // Printing After moving node
```

```
printf(" First List is: \n");
display(list1);
printf("\n Second list is: \n");
display(list2);
}
```

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

Sol:-

```
#include<stdio.h>
int top=-1;
#define size 20
int stack[20];
void push(int n)
{
    if(top==size-1)
       printf("Stack is Full");
    top++;
    stack[top]=n;
}
int pop()
{  int n;
   if(stack[top]==-1)
   {
      printf("\n Stack is Empty");
   }
   n=stack[top];
   top--;
   return n;
}
void main()
{
   int f,i,n,x,y,k,sum=0,cnt=1;
   printf("Enter number of elements in stack:");
   scanf("%d",&n);
   for(i=1; i<=n; i++)
   {    printf("Enter element:");
        scanf("%d",&x);
        push(x);
   }
```

```
printf(" Enter sum value:");
scanf(" %d", &k);
for(i=1; i<=n; i++)
{
    y = pop();
    sum += y; // y
    cnt += 1;
    if(sum == k)
    { int j;
      for(j=0; j<count; j++)
          printf(" %d ", stack[j]);
      f = 1;
      break;
    }
    push(y);
}
if(f != 1)
   printf(" Sum not found ");
}
```

Output:

```
Enter number of Elements in stack: 3
Enter element: 1
Enter element: 2
Enter element: 3
Enter sum value: 3
1 2
```

Outputs: (All outputs except 3rd one)

1.)

1. Insert
2. Delete
3. Display
4. Exit

Enter choice: 1

Enter position to insert: 1

Enter Data: 2

Enter choice: 1

Enter position to insert: 2

Enter Data: 3

Enter choice: 2

Enter position to delete: 2

Enter choice: 3

Linke List is 2

Enter choice: 4

Program Exit

2) (Output)

Enter number of elements to insert for list1: 3
Enter element to insert: 1
Enter element to insert: 2
Enter element to insert: 3
First list is:

3 → 2 → 1

Enter number of elements to insert for list2: 3
Enter element to insert: 4
Enter element to insert: 5
Enter element to insert: 6
Second List is:

4 → 5 → 6

After Merging:

3 → 6 → 2 → 5 → 1 → 4

4.) (Output)

1. Enqueue
2. Dequeue
3. Display & Reverse
4. Displayalternate
5. Exit

Enter your choice: 1

Enter element to insert: 3

Enter your choice: 1

Enter element to insert: 4

Enter your choice: 1

Enter element to insert: 5

Enter your choice: 3

5 4 3

Enter your choice: 4

3 5

Enter your choice: 5

Program Exit

5) (ii) Output

```
Enter number of nodes in list1: 3
Enter element: 1
Enter element: 2
Enter element: 3
Enter number of nodes in list2: 2
Enter element: 4
Enter element: 5
First List is:
4 -> 3 -> 2 -> 1
Second List is:
5
```

(After Moving first node)