Assignment-4

B. Jayanth
AP19110010394
CSE-F

1. Write a Program to insert and delete an element at the nth and kth position in a linked List where n and k is taken from user

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
        int n;
        struct node *next;
};
struct node *curr, *temp;
struct node *create(struct node*);
void inpos(struct node*);
void delpos(struct node*);
void main(void)
{
struct node *s;
int ch;
s=NULL;
do
{
printf("\n1. Create\n");
printf("2. Inpos\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: inpos(s);
                break;
```

```
                case 3 : delpos(s);
                         break;
}while(ch!=4)
}
struct node* create(struct node* x)
{
        if(x=NULL)
        {
            x=(struct node*)malloc(sizeof(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=(structnode*)malloc(sizeof(
                                        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 Inpos
3 delpos
4. Exit
Enter the choice 1
Enter the number 23
1 Create
2 Inpos
3 delpos
4. Exit
```

Enter the choice2
Enter the pos to be inserted: 2
Enter the number: 45
1. Create
2. Inpos
3. delpos
4. Exit
Enter the choice3
enter the pos to be deleted: 1

2. 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>
#include<assert.h>
struct node
{
    int data;
    struct node* next;
};
void Movenode(struct node** x; struct node** y);
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** x, struct node** y)
{
    struct node* newnode = *y;
    assert(newnode != NULL);
    *y = newnode->next;
    newnode->next = *x;
    *x = newnode;
}
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->next = (*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;
    push(&a, 1);
    push(&a, 2);
    push(&a, 3);
```

```
push(&b, 4);
push(&b, 5);
push(&b, 6);
res = SortedMerge(a, b);
printf("Merged linked List is: \n");
printList(res);
return 0;
```

3

Output:-

Merged linked List is

1 4 2 5 3 6

3 Find all the elements in the stack whose sum is equal to k

```
#include<stdio.h>
int s1[10], top1=-1, s2[10], top2=-1;
int s1empty()
{
    if(top1==-1)
        return 1;
    else
        return 0;
}
int s1top()
{
    return s1[top1];
}
int s1pop()
{
    top1--;
}
int s1push(int x)
{
    s1[++top1]=x;
}
int s2empty()
{
    if(top2==-1)
        return 1;
    else
        return 0;
}
int s2top()
{
    return s2[top2];
}
```

```
int s2pop()
{
    top2--;
}
int s2push(int x)
{
    s2[++top2]=x;
}
int sum(int k)
{
    int x;
    while(s1empty()!=1)
    {
        x=s1top();
        s1pop();
        while(s1empty()!=1)
        {
            if(x+s1top()==k)
            {
                printf("(%d %d)\n", x, s1top());
            }
            s2push(s1top());
            s1pop();
        }
        while(s2empty()!=1)
        {
            s1push(s2top());
            s2pop();
        }
    }
}
int main()
{
```

```
int n,i,e,k;
printf("enter the no of elements of stack:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
    scanf("%d",&e);
    s1push(e);
}
printf("enter the value of constant sum:")
scanf("%d",&k);
printf("The combinations whose sum is equal
                to k is:\n");
sum(k);
}
```

Output:-

enter the no of elements of stack: 4
enter the value of constant sum: 10
The combinations whose sum is equal to k is
1
2
3
4

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("1. Insertion\n2. Deletion\n3. Print
            Reverse\n4. Print Alternate\n5.
                Exit");
        printf("\nEnter your choice: ");
        scanf("%d", &choice);
        switch(choice){
case 1: printf("Enter the value to be insert: ");
        scanf("%d", &value);
        insert(value);
        break;
case 2: delete();
        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;
case4:printf("Alternate elements of the queue are:");
      for(int i=0;i<SIZE;i+=2)
      {
          if(queue[i]==0)
          continue;
          printf("\n%d",queue[i]);
      }
          break;
case5:exit(0);
default:printf("\nWrong selection!!! Try again!!!");
      }
}
void insert(int value)
{
    if(f==0 && r==SIZE-1)||f==r+1)
        printf("\nQueue is full!!! Insertion is not
                                    possible!!!");
    else{
        if(f==-1)
f=0;
        r=(r+1)%SIZE;
        queue[r]=value;
        printf("\nInsertion success!!!");
}
void delete(){
    if (f==-1)
        printf("\nQueue is Empty!!! Deletion is not
                                    possible!!!");
    else{
        printf("\nDeleted: %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:100
Insertion success!!!
1.Insertion
2.Deletion
3.Print Reverse
4.Print Alternate
Enter your choice:1
Enter the value to be insert:200
Insertion success!!!
1.Insertion
2.Deletion
3.Print Reverse
4.Print Alternate
Enter the value to be insert:300
Insertion success!!!
1.Insertion
2.Deletion
3.Print Reverse
4.Print Alternate
Enter your choice:3
The reversed queue is: 300 200 100
```

1 Insertion
2 Deletion
3 Print Reverse
4. Print Alternate
Enter your choice 4
Alternate elements of the queue are: 100 300

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

The major difference b/w Array and Linked List regards to their structure. Arrays are index based data structure where each element associated with an index. On the other hand, Linked list relies on references to the previous and next element.

(ii) Write a program to add the first element of one list to a another list for example we have {1,2,3} in list1 and {4,5,6} in list 2 we have to get {4,1,2,3} as output for list1 and {5,6} for list2

```
#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(sizeof(
                                    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 *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_curr;
        p_curr=p_next;
        q_curr=q_next;
    }
    *q=q_curr;
}
int main()
{
    struct node *p=NULL, *q=NULL;
    push(&p,1);
    push(&p,2);
    push(&p,3);
    printf("First linked List:\n");
    printList(p);
    push(&q,4);
    push(&q,5);
    push(&q,6);
    printf("Second linked list:\n");
    printList(q);
    merge(p,&q);
    printf("Modified First linked list:\n");
    printList(p);
    printf("Modified Second linked list:\n");
    printList(q);
    getchar();
    return 0;
}
```

Output :-

First linked List:

1 2 3

Second linked List:

4 5 6

Modified First linked list:

4 1 2 3

Modified second linked list:

5, 6