ASSIGNMENT-4

P. Kundhana Harshitha
AP19110010377
CSE-H

Q1 Write a program to insert and delete and element at the n^{th} and k^{th} position in a linked list where n and k is taken from user.

```
#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("\nEnter the element to insert:");
    scanf("%d",&newnode->data);
    if(n==1){
        newnode->next=head;
        head=newnode;
    }else{
        struct node* temp=head;
        for(i=1;i<n-1;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 (temp 1);
}else{
    int i;
    for (i=1; i<k-1; i++)
        temp1 = temp1 → next;
    struct node * temp 2 = temp1 → next;
    temp → next = temp 2 → next;
    free (temp 2);
}
}
void display (){
struct node * newnode;
newnode = head;
printf ("Linked list is :");
while (newnode != NULL){
    printf ("\n %d", newnode → data);
    newnode = newnode → next;
}
}
void main(){
int n, k, choice;
while (1){
    printf ("Enter 1- insertion 2- deletion 3-display
             4-exit of your choice : ");
    scanf ("%d", & choice);
    switch (choice){
      case 1:
          printf ("Enter nth position to insert : ");
          scanf ("%d", & n);
          insert(n);
          break;
```

```
        case 2:
            printf("Enter kth position to delete:");
            scanf("%d", &n);
            delete(n);
            break;
        case 3:
            display();
            break;
        case 4;
            exit(0);
        default: printf("Wrong input! Enter number
                        between 1-4");
        }
    }
}
```

Output:

```
Enter 1-insertion 2-deletion 3-display 4-exit of your choice:1
Enter nth position to insert : 1
Enter the element to insert : 1
Enter 1-insertion 2-deletion 3-display 4-exit of your choice: 1
Enter nth position to insert : 2
Enter the element to insert : 2
Enter 1-insertion 2-deletion 3-display 4-exit of your choice:1
Enter nth position to insert : 3
Enter the element to insert : 3
Enter 1-insertion 2-deletion 3-display 4-exit of your choice: 2
Enter kth position to delete : 2
Enter 1-insertion 2-deletion 3-display 4-exit of your choice: 1
Enter nth position to insert : 2
Enter the element to insert : 4
```

Enter 1- insertion 2-deletion 3- display 4-exit of your choice : 3

Linked list is :

1

4

3

Enter 1-insertion 2-deletion 3-display 4-exit of your choice : 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>
# include< malloc.h>
struct node{
        int data;
        struct node * next;
};
void display (struct node * head){
        struct node * temp = head;
        while (temp){
            printf("%d"→ ", temp → data);
            temp= temp→next;
        }
}
void insert (struct node ** head, int data){
        struct node * new node = (struct node*) malloc (size of
                                 (struct node));
        newnode→ data = data;
        newnode → next = * head;
        * head = new node;
```

```
}
void merge (struct node **a, struct node **b){
    struct node temp;
    struct node * tail = &temp;
    temp.next = NULL;
    while(1){
        // empty list cases
        if (*a == NULL){
            tail->next = NULL;
            break;
        }else if (*b == NULL){
            tail->next = *a;
            break;
        }else{
            tail->next = *a;
            *a = (*a)->next;
            tail->next = *b;
            tail = *b;
            *b = (*b)->next;
        }
    }
    *a = temp.next;
}
void main(){
    int i;
    struct node *list1 = NULL, *list2 = NULL;
    for(i=3; i>0; i--)
        insert(&list1, i);
    for(i=6; i>=4; i--)
        insert(&list2, i);
    printf("First List: ");
    display(list1);
```

```
        printf("\n Second List:");
        display(list2);
        merge(&list1, &list2);
        printf("\n After Herging the lists: \n");
        printf("New list is ");
        display(list1);
}
```

Output:

First list: 1 → 2 → 3 →
Second list: 4 → 5 → 6 →
After Herging the lists:
New list is: 1 → 4 → 2 → 5 → 3 → 6 →

Q3: Find all the elements in the stack whose sum is equal to k.

```
# include <stdio.h>
# include <limits.h>
# define max 1000
typedef struct STACK {
        int ar[max];
        int top;
} stack;
void push(stack *s, int data) {
        if (s -> top >= max-1) {
                return;
        }
        s -> top++;
        s -> ar[s -> top] = data;
}
int pop(stack *s) {
        if(s -> top < 0) {
                return INT_MIN;
        }
```

```
    s->top--;
    return s->ar[s->top+1];
}
void display(stack *s){
    int i;
    for(i=s->top; i>-1; i--){
        printf("%d", s->ar[i]);
    } printf("stack end\n");
}
int main(int argc, char constant *argv[]){
    stack s1;
    s1.top = -1;
    int expected, i,j, k, sum,n, num;
    printf("Enter the number of elements in stack:");
    scanf("%d", &n);
    while(n--){
        printf("Enter number: ");
        scanf("%d", &num);
        push(&s1, num);
    }
    printf("Enter expected value (sum): ");
    scanf("%d", &expected);
    while((i= pop(&s1)) != INT_MIN){
        if (i== expected)
            printf("%d\n", i);
        else{
            stack s2 = s1;
            while((j= pop(&s2)) != INT_MIN){
                sum = i+j;
                stack s3 = s2;
                stack store; store.top = -1;
```

```
            push (&store, i);
            push (&store, j);
            while ((s3.top != -1) || (expected >= sum)){
                if (sum == expected){
                    display(&store);
                    break;
                }
                int temp;
                if ((temp = pop(&s3)) == INT_MIN) break;
                sum += temp;
                push(&store, temp);

            }
        }
    }
    return 0;
}
```

Output:

```
Enter the number of elements in stack: 4
Enter number: 1
Enter number: 2
Enter number: 2
Enter number: 3
Enter expected value (sum): 4
1 3 stack end
2 2 stack end
```

Q4) Write a program to print the elements in a queue

i) in reverse order

```
# include <stdlib.h>
# include <stdio.h>
int queue[100], max, front=-1, rear=-1;
int value, choice;
    void enqueue(int);
    void dequeue();
    void display();
void main(){
    printf("Enter number of elements: ");
    scanf("%d", &max);
while(1){
    printf("\n Enter your choice 1-Insertion 2-deletion
            3-display 4-exit:");
    scanf("%d", &choice);
    switch(choice){
    case 1:
     printf("\n Enter the value of element to be
                                    inserted: ");
     scanf("%d", &value);
     enqueue(value);
     break;
    case 2:
     dequeue();
     break;
     case 3:
     display();
     break;
     case 4:
     exit(0);
     break;
    default:
```

```
        printf ("Wrong  selection ");
    }}}
void  enqueue (value) {
      if (rear = = max-1) {
          printf("Overflow ");
      }else {
         if (front = = -1) {
           front = 0;
         }rear++;
          queue [rear]= value;
        printf ("Element  is successfully  inseted ");
      }
}
void  dequeue ( ) {
      if (front = = rear) {
          printf ("Underflow");
       }else {
          printf (" Deletion  element  is %d", queue [front]);
          front++;
          if ( front = = rear + 1) {
               front = rear = -1;
          }
      }
}
void  display ( ) {
      if (rear = = -1) {
          printf ("Stack is  null ");
      }else {
        int i;
        for (i= rear ; i>=front ; i--) {
          printf ("%d \t", queue [i]);
```

```
    }}
}
```

(ii) in alternate order

```
#include<stdio.h>
#include<stdlib.h>
int queue[100], max, front=-1, rear=-1;
int value, choice;
void enqueue(int);
void dequeue();
void display();
void main(){
    printf("Enter the number of elements: ");
    scanf("%d", &max);
    while(1){
     printf("Enter your choice 1-Insertion 2-deletion
                              3-display  4-exit: ");
     scanf("%d", &choice);
     switch(choice){
          case 1:
          printf("Enter the value to be inserted: ");
          scanf("%d", &value);
          enqueue(value);
          break;
          case 2:
          dequeue();
          break;
          case 3:
          display();
          break;
```

```
        case 4:
            exit(0);
            break;
        default:
            printf("Wrong selection!");
}}}
void enqueue(value){
    if(rear==max-1){
        printf("overflow");
    }else{
        if(front==-1){
            front=0;}
            rear++;
            queue[rear]=value;
            printf("Element is inserted");
    }
}
void dequeue(){
    if(front==rear){
        printf("Underflow");
    }else{
        printf("Deletion element is %d", queue[front]);
        front++;
        if(front==rear+1){
            front=rear=-1;
        }
    }
}
void display(){
    if(rear==-1){
        printf("stack is null");
```

```
    }else{
        int i;
        for(i= front; i>= rear; i= i+2){
            printf("%d\t", queue[i]);
        }}
}
```

Output (i)

```
Enter the number of elements : 3
Enter your choice 1- insertion 2- deletion 3- display
                  4- exit: 1
Enter the value of element to be inserted : 1
Element is successfully inserted
Enter your choice 1- insertion 2- deletion 3- display 4exit: 1
Enter the value of element to be inserted : 2
Element is successfully inserted
Enter your choice 1- insertion 2- deletion 3- display 4-exit: 1
Enter the value of element to be inserted: 3
Element is successfully inserted
Enter your choice 1- insertion 2- deletion 3- display
                  4- exit: 3

3   2   1
Enter your choice 1- insertion 2- deletion 3- display
                  4- exit: 4
```

Output (ii)

```
It is same as 4(i) but display changes
Enter your choice 1- insertion 2- deletion 3- display
                  4- exit: 3

1   3
```

Enter your choice 1- Insertion 2- deletion 3- display
4- exit: 4

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

Array	Linked list
1) It is a collection of similar data types.	1) It is a collection of unordered linked elements known as nodes.
2) Accessing a element in an array is fast.	2) Linked list takes a bit slow time.
3) They are fixed size.	3) They are dynamic & flexible.
4) Elements are stored consecutively	4) They are stored randomly.
5) Position of the element is known by index	5) In linked list we have to start with head.

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

```
#include <stdio.h>
#include<stdlib.h>
struct node{
        int data;
        struct node *link;
}*head1 = NULL, *temp, *temp1, *head2 = NULL;
struct node * insert (struct node * head, int x){
    temp= (struct node *) malloc(sizeof(struct node));
    temp -> data = x;
    temp -> link = NULL;
```

```
if (head == NULL){
    head = temp;
}else{
    temp1 = head;
    while (temp1 -> link != NULL){
        temp1 = temp1 -> link;
    }temp1 -> link = temp;
}
return head;
}
int main(){
    int p, q, x, i;
    printf("Enternodelements in 1st linked list: ");
    scanf("%d", &p);
    for (i=0; i<p; i++){
        printf("Enter the element: ");
        scanf("%d", &x);
        head1 = insert(head1, x);
    }
    printf("Enternodelements in 2nd linked list: ");
    scanf("%d", &q);
    for (i=0; i<q; i++){
        printf("Enter the element: ");
        scanf("%d", &x);
        head2 = insert(head2, x);
    }
    temp = (struct node*) malloc (size of (struct node));
    temp -> link = head1;
    temp -> data = head2 -> data;
    head1 = temp;
```

```
    head2 = head2 → link;
    temp1 = head1;
    while (temp1 != NULL) {
        printf("%d ", temp → data);
        temp1 = temp1 → link;
    }
    printf("\n linked list 2 \n");
    temp1 = head2;
    while(temp1 != NULL) {
        printf("%d ", temp1 → data);
        temp1 = temp1 → link;
    }
}
```

Output:

```
Enter no of elements in 1st linked list: 3
Enter the element : 1
Enter the element : 2
Enter the element : 3
Enter no of elements in 2nd linked list: 3
Enter the element : 4
Enter the element : 5
Enter the element : 6
linkedlist 1:
4 1 2 3
linked list 2:
5 6
```