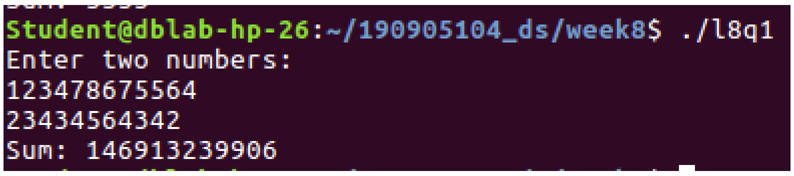
Week 8

1)

```
// Add two long positive integers represented using circular doubly linked list with header node.

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

typedef struct node * nodeptr;

typedef struct node{
	nodeptr rlink, llink;
	int data;
}node;

nodeptr create(){
	nodeptr temp = malloc(sizeof(node));
	return temp;
}

void push(nodeptr *n,int x){
	if(*n == NULL){
		*n = create();
		(*n)->data = x;
		(*n)->llink = (*n)->rlink = *n;
	}
	else{
		nodeptr temp = *n;
		while(temp->llink != *n)
			temp = temp->llink;

		nodeptr newnode = create();
		newnode->data = x;
		temp->llink = newnode;
		newnode->rlink = temp;
		newnode->llink = *n;
		(*n)->rlink = newnode;
	}
}

nodeptr read(){
	nodeptr head;
	char str[100];
	int i;
	scanf("%s",str);
	nodeptr n = create();
	n->llink = n->rlink = n;
	for(i=0;str[i];i++)
		push(&n,str[i]-'0');
	return n;
```

```
}

nodeptr add(nodeptr A, nodeptr B)
{
	int digit, sum, carry=0;
	nodeptr head,r,R,a,b;
	a=A->rlink;
	b=B->rlink;
	head = create();
	head->llink = head->rlink = head;
	while(a!=A && b!=B){
		sum = a->data + b->data +carry;
		digit = sum%10;
		carry = sum/10;
		push(&head,digit);
		a=a->rlink;
		b=b->rlink;
	}

	if(a!=A){
		r=a;
		R=A;
	}
	else{
		r=b;
		R=B;
	}
	while(r!=R){
		sum = r->data + carry;
		digit = sum%10;
		carry = sum/10;
		push(&head,digit);
		r = r->rlink;
	}

	if(carry)
		push(&head,carry);
	return head;
}

void display(nodeptr *n){
	for(nodeptr temp=(*n)->rlink; temp!=*n; temp=temp->rlink)
		printf("%d",temp->data);
	printf("\n");
}

int main()
{
	printf("Enter two numbers:\n");
	nodeptr A,B,sum;
	A = read();
	B = read();
	sum = add(A,B);
	printf("Sum: ");
	display(&sum);
	return 0;
}
```

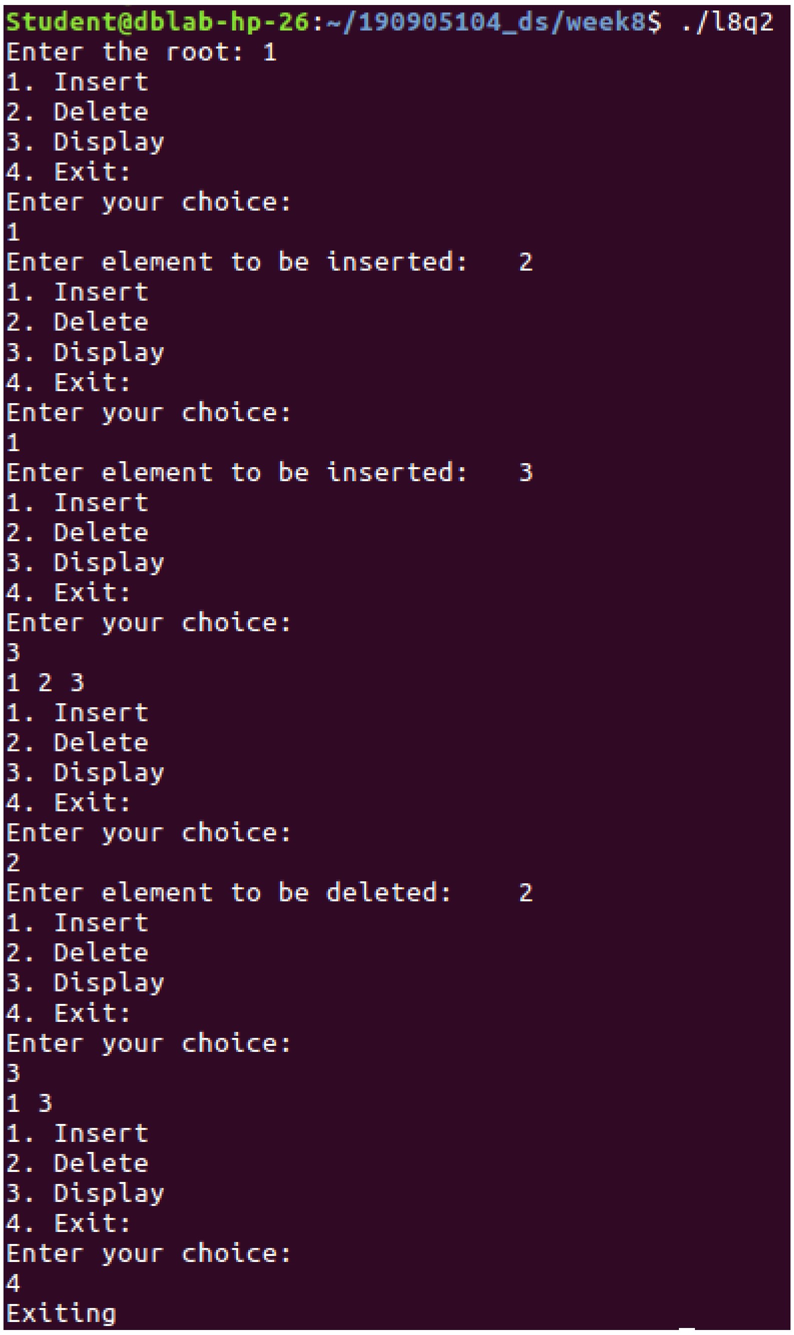


```
Student@dblab-hp-26:~/190905104_ds/week8$ ./l8q1
Enter two numbers:
123478675564
23434564342
Sum: 146913239906
```

2)

```
// Write a menu driven program to do the following using iterative functions:
// i) To create a BST for a given set of integer numbers
// ii)  To delete a given element from BST.
// iii) Display the elements using iterative in-order traversal.

#include <stdio.h>
#include <stdlib.h>
#define MAX 10

typedef struct node{
	int key;
	struct node *left, *right;
}*Node;

typedef struct{
	Node S[MAX];
	int tos;
}Stack;

Node newNode (int item){
	Node temp = (Node)malloc(sizeof(struct node));
	temp->key = item;
	temp->left = temp->right = NULL;
	return temp;
}

void push(Stack *s, Node n){
	s->S[++(s->tos)] = n;
}

Node pop(Stack *s){
	return s->S[(s->tos)--];
}

void inorder(Node root){
	Node curr;
```

```
        curr = root;
        Stack S;
        S.tos = -1;
        push(&S, root);
        curr = curr->left;
        while (S.tos != -1 || curr != NULL){
                while (curr != NULL){
                        push(&S, curr);
                        curr = curr->left;
                }
                curr = pop(&S);
                printf("%d ", curr->key);
                curr = curr->right;
        }
        printf("\n");
}

Node insert (Node node, int key){
        if (node == NULL)
                return newNode(key);
        if (key < node->key)
                node->left = insert(node->left, key);
        else if (key > node->key)
                node->right = insert(node->right, key);
        return node;
}

Node minValueNode(Node node){
        Node current = node;
        while (current && current->left != NULL)
                current = current->left;
        return current;
}

Node deleteNode(Node root, int key){
        if (root == NULL)
                return root;
        if (key < root->key)
                root->left = deleteNode(root->left, key);
        else if (key > root->key)
                root->right = deleteNode(root->right, key);
        else{
                if (root->left == NULL){
                Node temp = root->right;
                free(root);
                return temp;
                }
                else if (root->right == NULL){
                        Node temp = root->left;
                        free(root);
                        return temp;
                }
                Node temp = minValueNode(root->right);
                root->key = temp->key;
                root->right = deleteNode(root->right, temp->key);
        }
        return root;
}

int main(){
        Node root = NULL;
        int k;
        printf("Enter the root:\t");
        scanf("%d", &k);
        root = insert(root, k);
        int ch;
        do{
                printf("1. Insert\n2. Delete\n3. Display\n4. Exit:\n");
                printf("Enter your choice:\n");
                scanf("%d", &ch);
                switch (ch)
                {
                        case 1:
                        printf("Enter element to be inserted:\t");
                        scanf("%d", &k);
                        root = insert(root, k);
                        break;
                        case 2:
                        printf("Enter element to be deleted:\t");
                        scanf("%d", &k);
                        root = deleteNode(root, k);
                        break;
                        case 3:
                        inorder(root);
                        break;
                        case 4:
                        printf("Exiting\n");
                        break;
                        default:
                        printf("Invalid choice\n");
                }
        }while(ch != 4);
}
```



```
Student@dblab-hp-26:~/190905104_ds/week8$ ./l8q2
Enter the root: 1
1. Insert
2. Delete
3. Display
4. Exit:
Enter your choice:
1
Enter element to be inserted:   2
1. Insert
2. Delete
3. Display
4. Exit:
Enter your choice:
1
Enter element to be inserted:   3
1. Insert
2. Delete
3. Display
4. Exit:
Enter your choice:
3
1 2 3
1. Insert
2. Delete
3. Display
4. Exit:
Enter your choice:
2
Enter element to be deleted:    2
1. Insert
2. Delete
3. Display
4. Exit:
Enter your choice:
3
1 3
1. Insert
2. Delete
3. Display
4. Exit:
Enter your choice:
4
Exiting
```