WEEK 8

Lab Exercises:

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

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct node * nodeptr;
typedef struct node
{
nodeptr rlink, llink;
int data;
}node;
nodeptr create()
{
nodeptr temp = malloc(sizeof(node));
return temp;
}
void insert(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 readlong()
{
```

```
nodeptr head;
char str[100];
int i;
printf("Enter the integers you would like to ADD: ");
scanf("%s",str);
nodeptr n = create();
n->llink = n->rlink = n;for(i=0;str[i];i++)
insert(&n,str[i]-'0');
return n;
}
nodeptr addlong(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;
insert(&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;
insert(&head,digit);
r = r->rlink;
}
if(carry)
insert(&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()
{
nodeptr A,B,sum;
A = readlong();
B = readlong();
sum = addlong(A,B);
printf("Sum : ");
display(&sum);
return 0;
}
```

Test Case:

```
Enter the integers you would like to ADD: 762
Enter the integers you would like to ADD: 954
Sum : 1 7 1 6

Process returned 0 (0x0)   execution time : 10.080 s
Press ENTER to continue.
```

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.

Code:

```
#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;
}
}
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;
}
void main()
{
NODE root = NULL;
int k;
printf("Enter the root: ");
scanf("%d", &k);
root = insert(root, k);
int ch;
do
{
printf("\nEnter your choice:");
printf("\n1. Insert\n2. Delete\n3. Display\n4. Exit:\n");
scanf("%d", &ch);
switch (ch)
{
case 1:
printf("Enter element to be inserted: ");
scanf("%d", &k);
root = insert(root, k);
break;
case 2:
printf("Enter element to be deleted: ");
scanf("%d", &k);
root = deleteNode(root, k);
break;
```

```
case 3:
inorder(root);
break;
}} while (ch < 4);}
```

Test Case: