WEEK 8 LAB 8 :

1) Add two long positive integers represented using circular doubly linked list

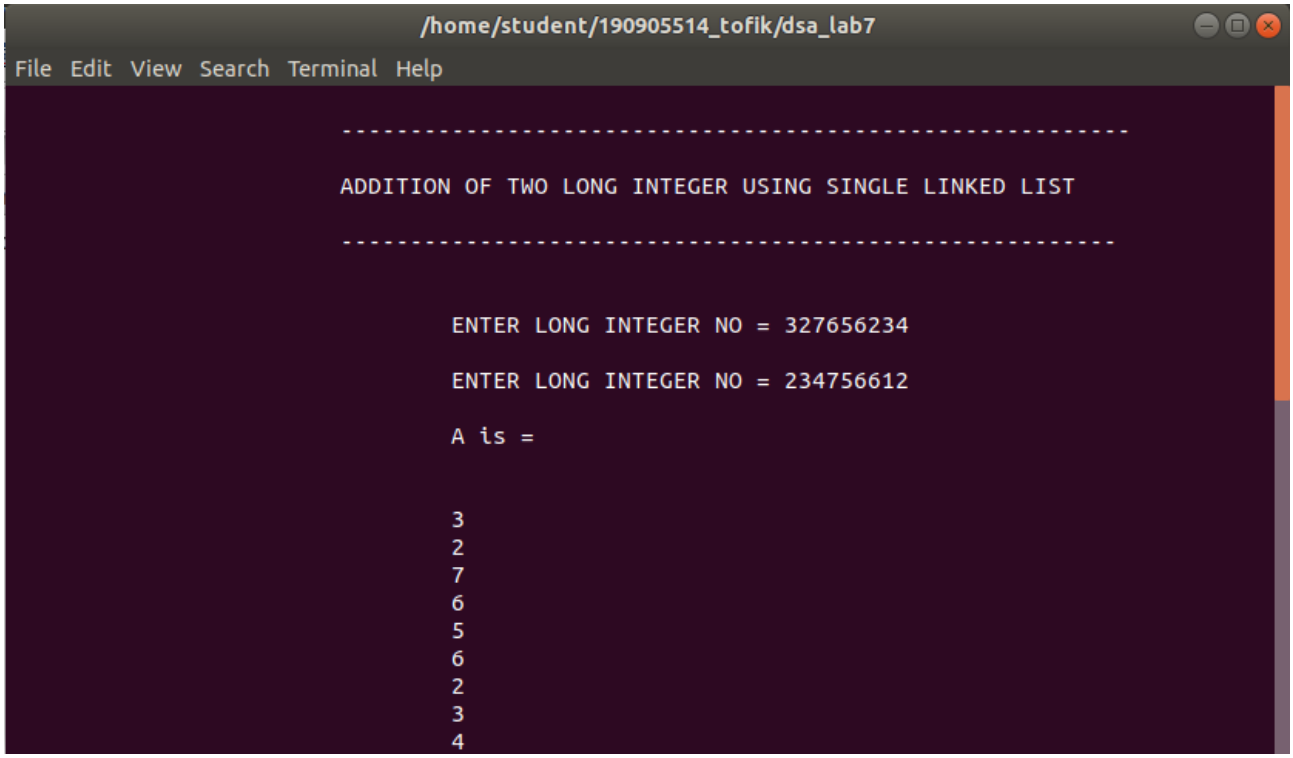
with header node.

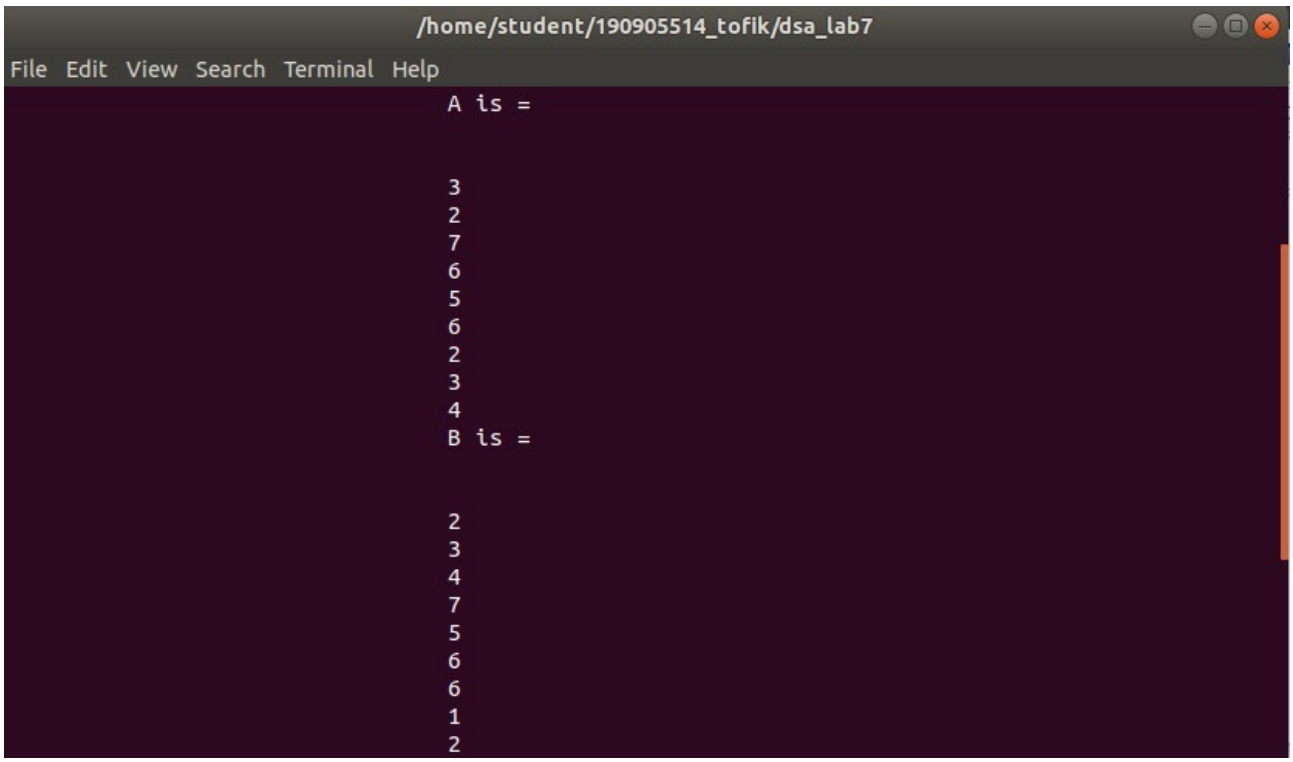
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct node *NODEPTR;
struct node
{
int info;
NODEPTR rlink;
NODEPTR llink;
};
NODEPTR getNode()
{
NODEPTR temp;
temp = (NODEPTR)malloc(sizeof(struct node));
if (temp == NULL)
{
printf("\n\t\t\t\tNO MEMEORY\n\n");
exit(0);
}
return temp;
}
NODEPTR insertfront(NODEPTR head, int data)
{
NODEPTR temp = getNode();
temp->info = data;
NODEPTR x = head->rlink;
temp->rlink = x;
head->rlink = temp;
x->llink = temp;
temp->llink = head;
return head;
}
NODEPTR readlongint()
{
NODEPTR head = getNode();
head->llink = head->rlink = head;
char s[100];
int i, n;
printf("\n\t\t\t\tENTER LONG INTEGER NO = ");
scanf("%s", s);
n = strlen(s);
for (i = n - 1; i >= 0; i--)
{
head = insertfront(head, s[i] - '0');
}
return head;
}
void display(NODEPTR head)
{
NODEPTR p = head->rlink;
if (head->rlink == head)
{
printf("\n\t\t\t\tEmpty\n");
return;
```

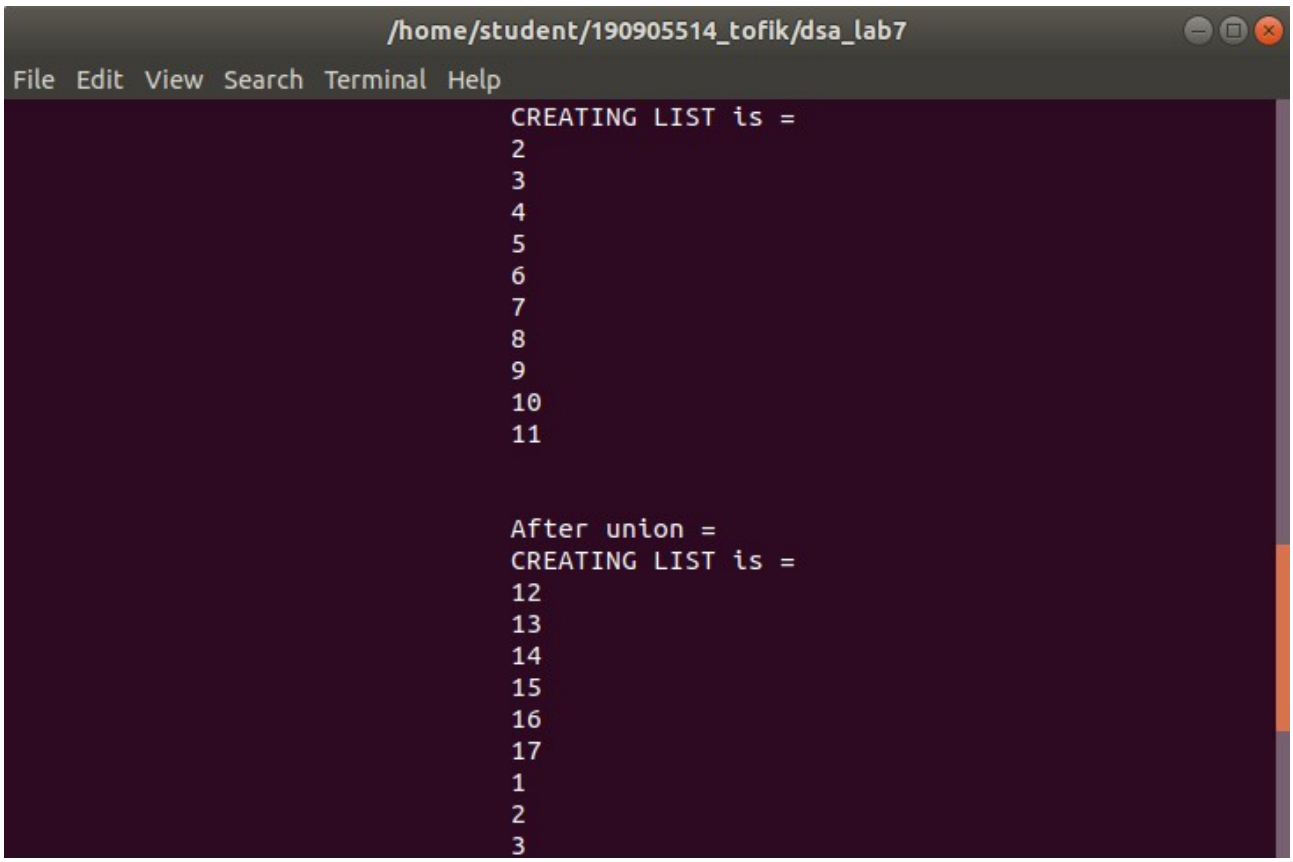
```
}
printf(" \n\n ");
while (p != head)
{
printf("\n\t\t\t\t%d", p->info);
p = p->rlink;
}

}
NODEPTR addlongint(NODEPTR a, NODEPTR b)
{
int x, y, z = 0;
NODEPTR s = getNode();
s->rlink = s->llink = s;
NODEPTR c = a;
NODEPTR d = b;
NODEPTR r, R;
a = a->llink;
b = b->llink;
while (c != a && d != b)
{
y = a->info + b->info + z;
x = y % 10;
s = insertfront(s, x);
z = y / 10;
b = b->llink;
a = a->llink;
}
if (a != c)
{
r = a;
R = c;
}
else
{
r = b;
R = d;
}
while (r != R)
{
y = r->info + z;
x = y % 10;
s = insertfront(s, x);
z = y / 10;
r = r->llink;
}
if (z != 0)
s = insertfront(s, z);
return s;
}
int main()
{
printf("\n\t\t\t----------------------------------------------------------\n");
printf("\n\t\t\tADDITION OF TWO LONG INTEGER USING SINGLE LINKED LIST \n");
printf("\n\t\t\t----------------------------------------------------------\n\n");
NODEPTR a = readlongint();
NODEPTR b = readlongint();
printf("\n\t\t\t\tA is = ");
display(a);
printf("\n\t\t\t\tB is = ");
display(b);
NODEPTR sum = addlongint(a, b);
printf("\n\t\t\t\tSUM is = ");
```

```
display(sum);
return 0;
}
```

OUTPUT :







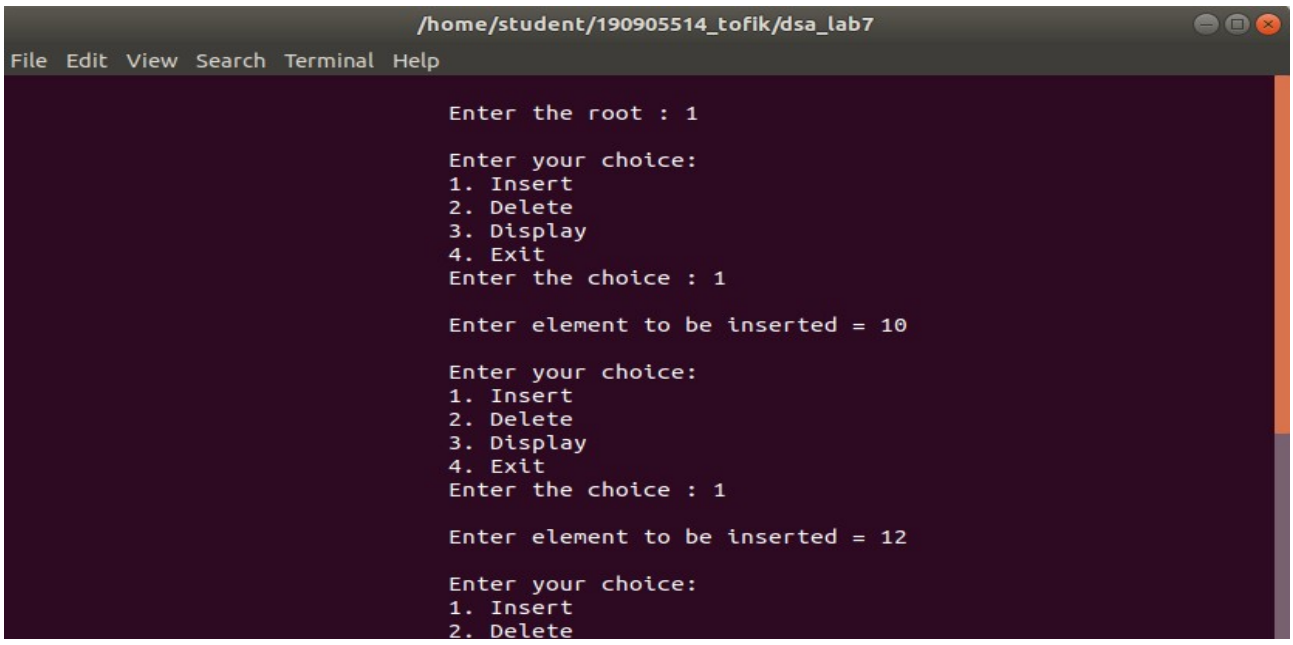
2) Write a menu driven program to do the following using iterative functions:
i) To create a BST for a given set of integer numbers
ii) To delete a given element from BST.
iii) Display the elements using iterative in-order traversal.

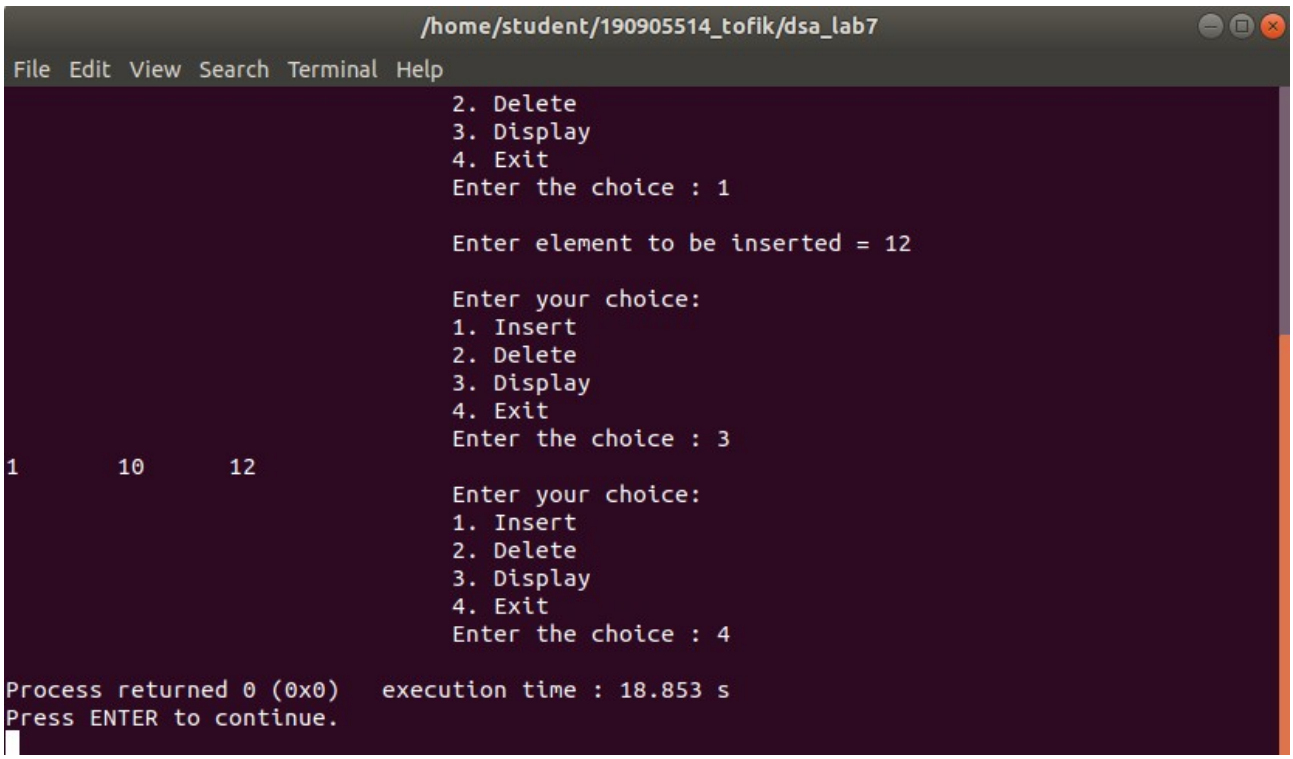
```
#include <stdio.h>
#include <stdlib.h>
#define MAX 10
typedef struct node
{
int key;
struct node *left, *right;
} * NODE;
typedef struct
{
NODE S[MAX];
int tos;
} STACK;
NODE newNODE(int item)
{
NODE temp = (NODE)malloc(sizeof(struct node));
temp->key = item;
temp->left = temp->right = NULL;
return 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)
```

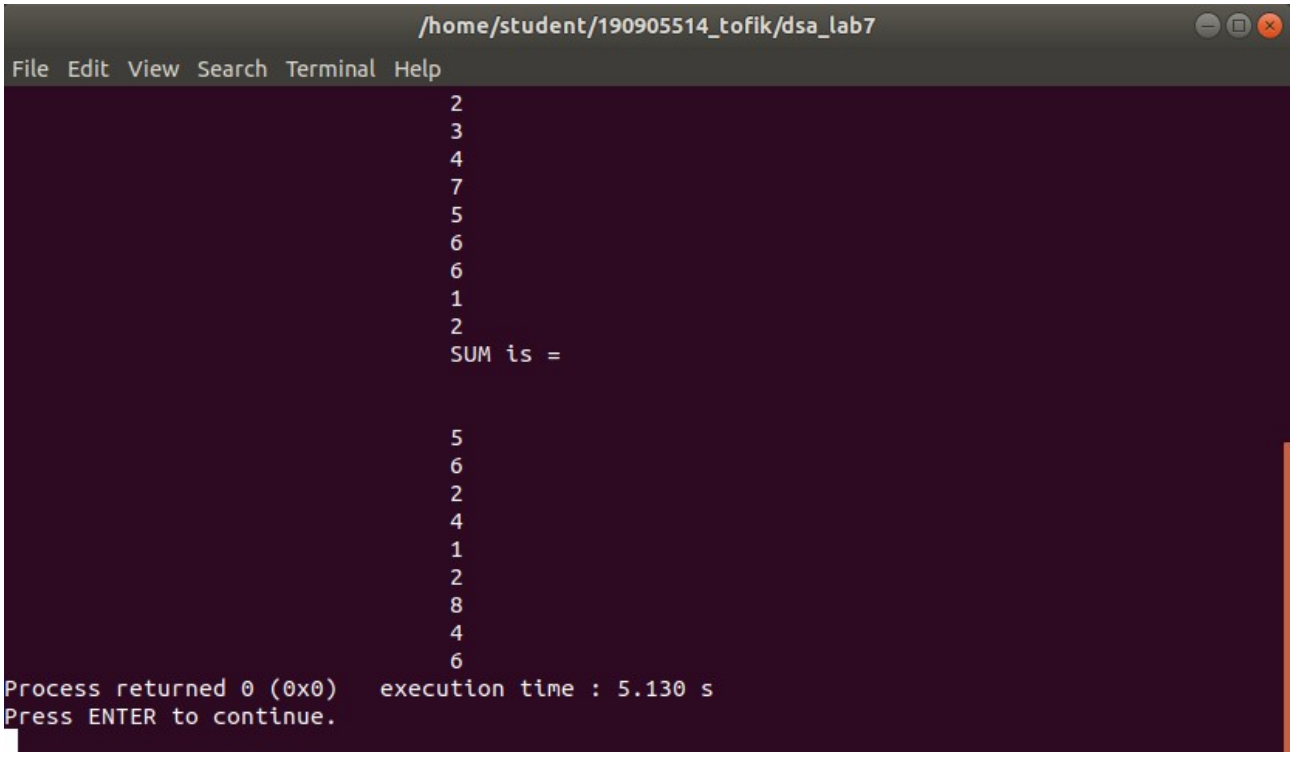
```
{
NODE curr;
curr = root;
STACK S;
S.tos = -1;
push(&S, root);
curr = curr->left;
while (S.tos != -1 || curr != NULL)
{
while (curr != NULL)
{
push(&S, curr);
curr = curr->left;
}
curr = pop(&S);
printf("%d\t", curr->key);
curr = curr->right;
}
}
NODE insert(NODE node, int key)
{
if (node == NULL)
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("\n\t\t\t\tEnter the root : ");
scanf("%d", &k);
root = insert(root, k);
int ch;
do
{
printf("\n\t\t\t\tEnter your choice:");
printf("\n\t\t\t\t1. Insert");
printf("\n\t\t\t\t2. Delete");
printf("\n\t\t\t\t3. Display");
printf("\n\t\t\t\t4. Exit");
printf("\n\t\t\t\tEnter the choice : ");
scanf("%d", &ch);
switch (ch)
{
case 1:
printf("\n\t\t\t\tEnter element to be inserted = ");
scanf("%d", &k);
root = insert(root, k);
break;
case 2:
printf("\n\t\t\t\tEnter element to be deleted = ");
scanf("%d", &k);
root = deleteNode(root, k);
break;
case 3:
inorder(root);
break;
}
} while (ch < 4);
}
```

OUTPUT :




/home/student/190905514_tofik/dsa_lab7
File Edit View Search Terminal Help
2. Delete
3. Display
4. Exit
Enter the choice : 1
Enter element to be inserted = 12
Enter your choice:
1. Insert
2. Delete
3. Display
4. Exit
Enter the choice : 3
1 10 12
Enter your choice:
1. Insert
2. Delete
3. Display
4. Exit
Enter the choice : 4
Process returned 0 (0x0) execution time : 18.853 s
Press ENTER to continue.


/home/student/190905514_tofik/dsa_lab7
File Edit View Search Terminal Help
2
3
4
7
5
6
6
1
2
SUM is =
5
6
2
4
1
2
8
4
6
Process returned 0 (0x0) execution time : 5.130 s
Press ENTER to continue.