VG101 Lab Worksheet

Lab 8

Instructor: Dr. Zhu Yifei
TA: Cao Hangrui
TA: Shan Haoxuan
TA: Wu Qinhang
TA: Xu Muchen

Code Quality Requirement

- Write codes in good indentation
- Make naming meaningful
- Include necessary comments in the code
- Split the code over functions if necessary
- Test the code as much as you can

Basic Exercise

Time Complexity

Determine the complexity of the following pieces of code, with respect to n.

Loops

Constant loop

```
for(int i=0;i<10;i++)
    cnt++;
```

Single loop

```
for(int i=0;i<n;i++)
    cnt++;
```

Nested loop I

```
for(int i=0;i<n;i++)
    for(int j=0;j<n;j++)
        cnt++;
```

Nested loop II

```
for(int i=0;i<n;i++)
    for(int j=i;j<n;j++)
        cnt++;
```

Nested loop III

```
for(int i=0;i<n;i++)
    for(int j=i;j<n;j++)
        for(int k=j;k<n;k++)
            cnt++;
```

Nested loop IV

```
for(int i=0;i<n;i++)
    for(int j=0;j*j<n;j++)
        cnt++;
```

Nested loop V

```
for(int i=0;i<n;i++)
    for(int j=0;i*j<n;j++)
        cnt++;
```

Nested loop VI

```
for(int i=0;i<n;i++)
    for(int j=0;j<n;j+=i)
        cnt++;
```

Nested loop VII

```
for(int i=n;i>0;i/=2)
    for(int j=0;j<i;j++)
        cnt++;
```

Nested loop VIII

```
for(int i=n;i>0;i/=2)
    for(int j=0;j<n;j++)
        cnt++;
```

Nested loop IV

```
for(int i=0;i<n;i++)
    for(int j=0,temp=rand()%n;j<temp;i++,j++)
        cnt++;
```

Recursion

Simple recursion I

```
int fac(int n)
{
    if (n<=1)
        return 1;
    return n*fac(n-1);
}
```

Simple recursion II

```
int fibo(int n)
{
    if (n<=1)
        return 1;
    return fibo(n-1)+fibo(n-2);
}
```

Simple recursion III

```
int Pow(int a,int n)
{
    if(n==0)
        return 1;
    int res=Pow(a,n/2);
    if(n%2==1)
        return res*res*a;
    else
        return res*res;
}
```

Binary Search

Implement the function `int count(int a[],int len,int element)`, which returns the number of occurence of `element`.

The array `a` has `n` integers, in ascending order.

Make sure the time complexity of your code is $O(\log len)$ in the worst case.

```c
#include<stdio.h>
int count(int a[], int len, int element)
{
    //TODO: count the number of occurence of element
    //      within time complexity O(log len).
}
int main()
{
    int a[] = {-1, 0, 1, 1, 1, 1, 2, 2, 3};
    printf("%d", count(a, 9, 1));          //output: 4
}
```

Practice Exercise

Arbitrary Precision Integer Arithmetic (JOJ)

Procedure Oriented Programming

Recall how long addition and long multiplication are performed. Implement the arbitray precision integer addition and multiplication in C with the code skeleton below.

Use this code to find the exact number of

1. The n-th Fibonacci number.
2. 2^m.

```c
#include <stdio.h>
#include <string.h>
#define maxn 2000
typedef struct BigInt
{
    int len;       //The length of the number (number of digits)
    int val[maxn]; //The value of the each digits. Index starts from 0
} BigInt;
int max(int a, int b) { return a > b ? a : b; }

void clear(BigInt *c)
{
    memset((void *)c, 0, sizeof(BigInt));
}

// print the struct in one line.
void print(BigInt a)
{
    for (int i = a.len - 1; i >= 0; i--)
        printf("%d", a.val[i]);
    puts("");
}

BigInt scan()
{
    BigInt a;
    clear(&a);
    char s[maxn];
    scanf("%s", s);
```

```
    int len = strlen(s);
    //TODO: convert the string s to BigInt
    return a;
}

// only available for INT_MIN<=x<=INT_MAX
BigInt construct(int x)
{
    BigInt big;
    clear(&big);
    //TODO: convert the string s to BigInt
    return big;
}

BigInt add(BigInt a, BigInt b)
{
    BigInt c;
    clear(&c);
    //TODO: return the BigInt of a+b
    return c;
}

BigInt mul(BigInt a, BigInt b)
{
    //TODO: imitate the previous function, return the BigInt of a*b
}

int main()
{
    BigInt a = construct(123456789), b = construct(987654321),
           c = construct(19260817), d = construct(998244353);
    print(a), print(b), print(c), print(d);
    print(add(a, b));                 //1111111110
    print(mul(a, b));                 //121932631112635269
    print(mul(c, d));                 //19227001804416401
    print(mul(mul(a, b), mul(c, d))); //2344398918419877713784957714646869
    print(mul(scan(), scan()));       //Test your self!
    return 0;
}
```

Simple double linked list (JOJ)

Follow the starter files released on canvas.

All you need to do is to submit a zip file containing a single `double_list.c` file (it should add #include"double_list.h" in the beginning) on JOJ.

Note that you **should NOT** submit `double_list.h`.

The header file `double_list.h` includes:

```
#ifndef LAB8NEW_DOUBLE_LIST_H
#define LAB8NEW_DOUBLE_LIST_H
typedef struct node{
    int data;
```

```
    // the next node and the previous node
    struct node *next;
    struct node *prev;
}node_t;
void push(node_t **head_ref,int num);
void insertAfter(node_t* prev_node, int num);
void append(node_t ** head_ref, int new_data);
void printList(node_t* node);
void deleteNode(node_t ** head_ref, node_t * del);
void freeList(node_t **head_ref);
#endif //LAB8NEW_DOUBLE_LIST_H
```

A sample double_list.c file:

```
//
// Created by Dell on 2020/7/3.
//

#include "double_list.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void push(node_t **head_ref,int num){
    //TODO: head reference is the reference to the node head, push one element to the linklist
        //Hint: notice if no element exist in double linked list
}

void deleteNode(node_t ** head_ref, node_t * del)
{

    //TODO: Implement function to delete a node in a Doubly Linked List.
    //   head_ref --> pointer to head node pointer.
    //   del  -->  pointer to node to be deleted

}

void insertAfter(node_t *prev_node, int num){
    //TODO: Insert List node after a given node.

}

void append(node_t ** head_ref, int new_data)
{
    //Given a reference (pointer to pointer) to the head of a DLL and an int, appends a new node at the end

}

void printList(node_t* node)
{

    printf("\nVisit Elements in forward direction \n");
    while (node != NULL) {
        printf(" %d ", node->data);
        node = node->next;
```

```
    }


}
void freeList(node_t **head_ref){
    // TODO: Free the list
}
```

Note: **do NOT** include the `main` function when submitting. However, you may need to include a main function to test your implementation locally on your own computer.