**BÀI TẬP VỀ KIỂM THỬ HỘP TRẮNG**

**Bài 1:** Cho hàm sau:

int linearSearch(int arr[], int target, int n)

{

for( int i = 0; i < n; i++ ) {

if ( arr[i] ==

target) {

return i;

}

}

return -1;

}

Để kiểm thử cho hàm trong chương trình trên, hãy thực hiện các yêu cầu sau đây:

* Xây dựng đồ thị luồng điều khiển nhị phân cho hàm trên.
* Xác định độ phức tạp của đồ thị.
* Xác định các đường thi hành tuyến tính độc lập ứng với đồ thị luồng điều khiển.
* Xác định các ca kiểm thử cơ bản.
* Dùng Junit thực hiện các ca kiểm thử trên.

**Bài 2:** Cho hàm sau:

**public** **static** **void** selectionSort(**int**[] myArray) {

**int** n = myArray.length;

**for** (**int** i = 0; i < n - 1; i++) {

**int** minIndex = i;

// Find the index of the minimum element in the unsorted part of the array

**for** (**int** j = i + 1; j < n; j++) {

**if** (myArray[j] < myArray[minIndex]) {

minIndex = j;

}

}

// Swap the minimum element with the first element of the unsorted part

**int** temp = myArray[minIndex];

myArray[minIndex] = myArray[i];

myArray[i] = temp;

}

}

Để kiểm thử cho hàm trong chương trình trên, hãy thực hiện các yêu cầu sau đây:

* Xây dựng đồ thị luồng điều khiển nhị phân cho hàm trên.
* Xác định độ phức tạp của đồ thị.
* Xác định các đường thi hành tuyến tính độc lập ứng với đồ thị luồng điều khiển.
* Xác định các ca kiểm thử cơ bản.
* Dùng Junit thực hiện các ca kiểm thử trên.

**Bài 3:** Cho hàm sau:

***public******static******void*** *cal(****int*** *a[],* ***int*** *n,* ***int*** *b[],* ***int*** *m,* ***int*** *c[]) {*

***int*** *k = 0, i = 0, j = 0;* ***1***

***2******3***

***while*** *(i < n && j < m) {*

***if*** *(a[i] < b[j]) { 4*

*c[k] = a[i];* ***5***

*i++;*

*}* ***else*** *{*

*c[k] = b[j];* ***6***

*j++;*

*}*

*k++;* ***7***

*}*

***while*** *(i < n) {* ***8***

*c[k] = a[i];* ***9***

*i++;*

*k++;*

*}*

***while*** *(j < m) {* ***10***

*c[k] = b[j];* ***11***

*j++;*

*k++;*

*}*

*}*

Để kiểm thử cho hàm trong chương trình trên, hãy thực hiện các yêu cầu sau đây:

* Xây dựng đồ thị luồng điều khiển nhị phân cho hàm trên.
* Xác định độ phức tạp của đồ thị.
* Xác định các đường thi hành tuyến tính độc lập ứng với đồ thị luồng điều khiển.
* Xác định các ca kiểm thử cơ bản.
* Dùng Junit thực hiện các ca kiểm thử trên.

**Bài 4:**

Cho hàm sau:

***public******static******int*** *cal(****int*** *m1,* ***int*** *d1,* ***int*** *m2,* ***int*** *d2,* ***int*** *y) {*

***int*** *numDays;*

***if*** *(m2 == m1)*

*numDays = d2 - d1;*

***else*** *{*

***int*** *m4 = y % 4;*

***int*** *m100 = y % 100;*

***int*** *m400 = y % 400;*

***int****[] daysIn = { 0, 31, 0, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };*

***if*** *((m4 != 0) || ((m100 == 0) && (m400 != 0)))*

*daysIn[2] = 28;*

***else***

*daysIn[2] = 29;*

*numDays = d2 + (daysIn[m1] - d1);*

***int*** *i = m1 + 1;*

***while*** *(i <= m2 - 1) {*

*numDays = numDays + daysIn[i];*

*i++;*

*}*

*}*

***return*** *(numDays);*

*}*

Để kiểm thử cho hàm trong chương trình trên, hãy thực hiện các yêu cầu sau đây:

* Xây dựng đồ thị luồng điều khiển nhị phân cho hàm trên.
* Xác định độ phức tạp của đồ thị.
* Xác định các đường thi hành tuyến tính độc lập ứng với đồ thị luồng điều khiển.
* Xác định các ca kiểm thử cơ bản.
* Dùng Junit thực hiện các ca kiểm thử trên.

**Bài 5:**

**Cho hàm sau:**

int Sum(int a[], int n){

int S,i;

if(n==0)

return -1;

S=0;

i=0;

while(i<n){

S=S+a[i];

i++;

}

return S;

}

Để kiểm thử cho hàm trong chương trình trên, hãy thực hiện các yêu cầu sau đây:

* Xây dựng đồ thị luồng điều khiển nhị phân cho hàm trên.
* Xác định độ phức tạp của đồ thị.
* Xác định các đường thi hành tuyến tính độc lập ứng với đồ thị luồng điều khiển.
* Xác định các ca kiểm thử cơ bản.
* Dùng Junit thực hiện các ca kiểm thử trên.

**Bài 6:** Cho hàm sau:

int gcd(int a, int b){

if (a == 0 || b == 0){

return a + b;

}

while (a != b){

if (a > b){

a -= b; // a = a - b

}else{

b -= a;

}

}

return a;

}

Để kiểm thử cho hàm trong chương trình trên, hãy thực hiện các yêu cầu sau đây:

* Xây dựng đồ thị luồng điều khiển nhị phân cho hàm trên.
* Xác định độ phức tạp của đồ thị.
* Xác định các đường thi hành tuyến tính độc lập ứng với đồ thị luồng điều khiển.
* Xác định các ca kiểm thử cơ bản.
* Dùng Junit thực hiện các ca kiểm thử trên.

double average(int value, int step, int min, int max)

{

double res = 0.0;

int i=0; int k=0; int count =0; int sum =0;

double temp = 0.0;

while(k<step){

k++; temp =value \*k;

if(temp>min && temp<= max){

count ++;

sum +=temp;

}

if(temp > max ) break;

}

if (count >0 {res = sum/count;}

return res;

}