**Bài 1- (7 điểm): Hỏi bài tập lớn theo format**

**I- Tên đề tài**

Tìm hiểu công cụ kiểm thử tự động Selenium IDE và ứng dụng.

**II- Các yêu cầu**

1- **(3.0 điểm).** Quy trình hoạt động của công cụ Selenium IDE và ví dụ minh họa

2 - **(4.0 điểm).**  Ứng dụng công cụ Selenium IDE cho bài toán cụ thể

a. Viết các test case

b. Chạy demo ứng dụng cho các test case

**Bài 2 (3 điểm): Hỏi một câu về kiến thức kiểm thử theo hệ thống các câu hỏi dưới đây.**

***Câu 21 (bao gồm câu 1+2) : phân biệt kiểm thử hộp trắng và kiểm thử hộp đen***

|  |  |  |
| --- | --- | --- |
|  | Kiểm thử hộp đen | Kiểm thử hộp trắng |
| Khái niệm | -Là một phương pháp kiểm thử phần mềm mà việc kiểm tra các chức năng của một ứng dụng không cần quan tâm vào cấu trúc nội bộ hoặc hoạt động của nó. | Kiểm thử hộp trắng (While box test) là phương pháp thử nghiệm phần mềm, trong đó các thiết kế, cấu trúc giải thuật bên trong, và việc thực hiện các công việc đều được biết đến |
| Đối tượng kiểm thử | - Là thành phần phần mền (TPPM) có thể là 1 hàm chức năng, 1 modul chức năng, 1 phân hệ chức năng... | - Là 1 thành phần của phần mềm (1 chức năng, 1 module chức năng, 1 phân hệ chức năng....) |
| Phương pháp | -được áp dụng hầu như đến mọi cấp độ của kiểm thử phần mềm:  +Kiểm thử đơn vị (Unit test)  + Kiểm thử tích hợp (Intergration test)  + Kiểm thử hệ thống (System test)  + Kiểm thử chấp nhận (Acceptance test). | -Với những TPPM quá lớn sẽ tốn rất nhiều thời gian và công sức để kiểm thử nếu như dùng kiểm thử tích hợp (Integration test) hay kiểm thử chức năng (Functional test)).  -Kỹ thuật white box test thích hợp dùng để kiểm thử đơn vị (Unit test) |
| Đặc điểm | - dựa vào thông tin duy nhất là các đặc tả về yêu cầu chức năng của TPPM tương ứng  - Người kiểm thử không cần thiết phải có kiến thức về việc mã hoá, cấu trúc bên trong của TPPM, cũng như không yêu cầu phải biết lâp trình phần mềm.  - Việc kiểm thử được tiến hành dựa vào việc kiểm thử TPPM làm được gì, có phù hợp với yêu cầu của người dùng hay không. Các tester nhập số liệu vào phần mềm và chỉ cần xem kết quả của phần mềm và các mục tiêu kiểm tra.  - Mức test này thường yêu cầu các tester phải viết test case đầy đủ trước khi test; khi test, đơn giản chỉ cần thực hiện theo các bước mô tả trong test case thao tác và nhập data vào, sau đó xem kết quả trả về hoặc hành vi của phần mềm, rồi so sánh với kết quả mong đọi được viết trong testcase | -dựa vào giải thuật, cấu trúc bên trong chức năng của TPPM tương ứng.  -Người kiểm thử phải có kiến thức nhất định về việc mã hoá, cấu trúc bên trong của chức năng, biết lâp trình phần mềm.  -Việc kiểm thử được tiến hành dựa vào việc kiểm xem giải thuật, mã lệnh đã làm có đúng không.  -Mức test này thường yêu cầu các tester phải viết test case đầy đủ các nhánh trong code; khi test, sẽ set điều kiện và data để chạy vào đủ tất cả các nhánh trong giải thuật, đảm bảo thực hiện đầy đủ. |
| Tạo + thực hiện testcase | -Khi viết test case: Dựa vào yêu cầu và giao diện bên ngoài của chương trình (Không can thiệp vào bên trong code của chương trình)  -Khi thực hiện test: Thực hiện trên giao diện của chương trình (yêu cầu chương trình phải chạy được mới test được, không can thiệp vào code) | - Khi viết test case: Dựa vào yêu cầu và nội dung Source Code (can thiệp vào bên trong Code của chương trình)  - Khi thực hiện test: Thực thi test trong code (không cần thực thi chương trình, vì thực hiện test white box sẽ sử dụng framework nào đó hỗ trợ (Ví dụ như test kiểu debug) |
| VÍ DỤ: |  |  |

***Câu 3: thế nào là kiểm thử chức năng? Hãy nêu các bước chính của phương pháp kiểm thử chức năng bằng sơ đồ hình vẽ***

1. Khái niệm:

* Kiểm thử chức năng là một loại kiểm thử hộp đen (black box) và test case của nó được dựa trên đặc tả của ứng dụng phần mềm/thành phần đang test. Các chức năng được test bằng cách nhập vào các giá trị nhập và kiểm tra kết quả đầu ra, và ít quan tâm đến cấu trúc bên trong của ứng dụng.
* Nó là một qui trình cố gắng tìm ra các khác biệt giữa đặc tả bên ngoài của phần mềm và thực tế mà phần mềm cung cấp. Với các đặc tả bên ngoài của phần mềm là đặc tả chính xác về hành vi của phần mềm theo góc nhìn của người dùng.

1. **Các bước KTCN:**

***Câu 4: Thế nào là kiểm thử giá trị biên? Cho một ví dụ theo ý hiểu của anh/ chị.***

* là kỹ thuật thiết kế test case và hoàn thành phân vùng tương đương. Mục tiêu là lựa chọn các test case để thực thi giá trị biên vì lỗi thường xảy ra ở gần các giá trị biên này. Chương trình viết bằng ngôn ngữ không có kiểm tra kiểu mạnh càng cần kiểm thử giá trị biên Javascript, php, Visual Basic
* **VD: a <= x <=b thì sẽ chọn a, a+1, a+b/2, b-1, b.**

***Câu 5: Thế nào là kiểm thử lớp tương đương? Hãy áp dụng phương pháp kiểm thử lớp tương đương cho bài toán xét các trường hợp của tam giác?***

* Là phương pháp kiểm thử hộp đen chia miền đầu vào của một chương trình thành các lớp dữ liệu, từ đó suy dẫn ra các ca kiểm thử. Thiết kế ca kiểm thử cho KTTĐ dựa trên sự đánh giá về các vùng tương đương với một điều kiện vào. Thiết kế test case bằng phân vùng tương đương tiến hành theo hai bước:

+ Xác định các lớp tương đương

+Xác định các ca kiểm thử

* **ÁP DỤNG**

***Câu 6: Thế nào là kiểm thử bằng bảng quyết định? Nêu cấu trúc của một bảng quyết định.***

* Là phương pháp chính xác nhất trong các kỹ thuật kiểm thử chức năng. Bảng quyết định là phương pháp hiệu quả để mô tả các sự kiện, hành vi sẽ xảy ra khi một số điều kiện thỏa mãn.
* Cấu trúc:

+ Các biểu thức điều kiện C1, C2, C3...

+ Giá trị điều kiện T, F, -

+ Các hành động A1, A2, A3, A4...

+ Giá trị hành động, có xảy ra hay không. X là có

***Câu 7: Hãy nêu các tính chất để áp dụng phương pháp kiểm thử bằng bảng quyết định cho bài toán.***

* Để xác định các ca kiểm thử dựa trên bảng quyết định, ta dịch các điều kiện thành các đầu vào và các hành động thành các đầu ra. Đôi khi các điều kiện sẽ xác định các lớp tương đương của đầu vào và các hành động tương ứng với các mô-đun xử lý chức năng đang kiểm thử.
* Những bài toán phù hợp với bảng quyết định:
  + Chương trình có nhiều lệnh rẽ nhánh - nhiều khối If-Then-Else
  + Các biến đầu vào có quan hệ với nhau
  + Có quan hệ nhân quả giữa đầu vào và đầu ra
  + Có độ phức tạp Cyclomatic cao
* Bảng quyết định không dễ áp dụng cho các bài toán lớn (với n điều kiện có 2n quy tắc).

GIẢI VÀ BIỆN LUẬN PT BẬC 2

Program GIAI\_PHUONG\_TRINH\_BACII;

Uses crt;

Var a,b,c,d,x,x1,x2: real;

Begin

  Clrscr;

  Writeln('GIAI PHUONG TRINH BAC II: ');

  Writeln('-------------------------------------------');

  Write('Nhap he so a='); readln(a);

  Write('Nhap he so b='); readln(b);

  Write('Nhap he so c='); readln(c);

  If a=0 then

    If b=0 then

      If c=0 then

        Writeln('Phuong trinh co vo so nghiem')

      Else Writeln('Phuong trinh vo nghiem')

    Else Writeln('Phuong trinh co mot nghiem: x=',-c/b: 4: 2)

  Else

    Begin

      D:=b\*b-4\*a\*c;

      If D=0 then Writeln('Phuong trinh co nghiem kep: x=',-b/(2\*a): 4: 2)

      Else

        If D<0 then Writeln('Phuong trinh vo nghiem')

        Else

           Begin

             x1:=(-b-sqrt(D))/(2\*a);

             x2:=(-b+sqrt(D))/(2\*a);

             Writeln('Pt co 2 nghiem la x1= ',x1:4:2 ,' va x2=',x2:4:2);

           End;

     End;

Readln

End.

**Câu 8: Viết chương trình giải và biện luận phương trình bậc 2: a.x2 + b.x + c = 0 . Áp dụng kiểm thử giá trị biên để xây dựng bộ dữ liệu kiểm thử cho chương trình.**

**Kiểm thử giá trị biên**

Phương trình  có miền xác định là . Để nó là phương trình bậc hai thì *a* phải khác 0. Giả sử các giá trị *a*, *b*, *c* nằm trong đoạn . Ta có bảng kiểm thử các giá trị biên cho bài toán giải phương trình bậc hai như sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Case | *a* | *b* | *c* | Expected Output |
| 1 | 5 | 5 | 5 | Vô nghiệm |
| 2 | 5 | 5 | 4 | Vô nghiệm |
| 3 | 5 | 4 | 5 | Vô nghiệm |
| 4 | 4 | 5 | 5 | Vô nghiệm |
| 5 | 5 | 4 | 4 | Vô nghiệm |
| 6 | 4 | 5 | 4 | Vô nghiệm |
| 7 | 4 | 4 | 5 | Vô nghiệm |
| 8 | 5 | 5 | -5 | Có 2 nghiệm |
| 9 | 5 | -5 | 5 | Vô nghiệm |
| 10 | -5 | 5 | 5 | Có 2 nghiệm |
| 11 | 5 | -5 | -5 | Có 2 nghiệm |
| 12 | -5 | 5 | -5 | Vô nghiệm |
| 13 | -5 | -5 | 5 | Có 2 nghiệm |
| 14 | 4 | 5 | -5 | Có 2 nghiệm |
| 15 | 5 | 4 | -5 | Có 2 nghiệm |
| 16 | 5 | -5 | 4 | Vô nghiệm |
| 17 | -5 | 5 | 4 | Có 2 nghiệm |
| 18 | 4 | -5 | 5 | Vô nghiệm |
| 19 | 4 | -4 | 5 | Vô nghiệm |
| 20 | 4 | 5 | -4 | Có 2 nghiệm |
| 21 | 4 | 4 | -5 | Có 2 nghiệm |

**Câu 9: Viết chương trình giải và biện luận phương trình bậc 2: a.x2 + b.x + c = 0 . Áp dụng kiểm thử lớp tương đương để xây dựng bộ dữ liệu kiểm thử cho chương trình.**

* Miền đầu ra cho các giá trị:
* Không phải là phương trình bậc hai.
* Phương trình vô nghiệm.
* Phương trình có nghiệm kép.
* Phương trình có hai nghiệm phân biệt.
* Xác định các lớp tương đương:
* .
* 
* 
* 
* Bảng kiểm thử

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Case | *a* | *b* | *c* | Expected Output |
| 1 | 0 | 0 | 9 | Không phải là phương trình bậc hai |
| 2 | 1 | 1 | 1 | Phương trình vô nghiệm |
| 3 | 1 | -2 | 1 | Phương trình có nghiệm kép |
| 4 | 1 | -3 | 2 | Phương trình có hai nghiệm phân biệt: |

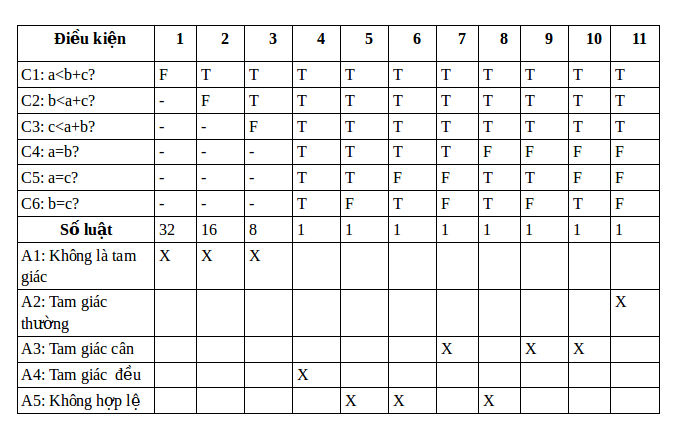
**Câu 10: Viết chương trình giải và biện luận phương trình bậc 2: a.x2 + b.x + c = 0 . Áp dụng kiểm thử bằng bảng quyết định để xây dựng bộ dữ liệu kiểm thử cho chương trình.**

1. **Xây dựng bảng quyết định cho bài toán giải phương trình bậc hai.**

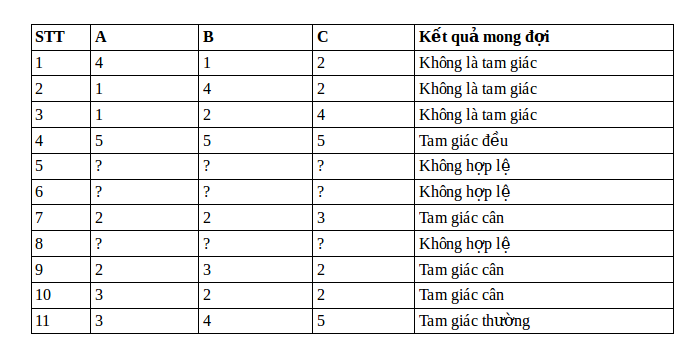
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Điều kiện | 1 | 2 | 3 | 4 |
| C1: | T | F | F | F |
| C2: | - | T | F | F |
| C3: | - | - | T | F |
| Số luật | 4 | 2 | 1 | 1 |
| A1: Không phải phương trình bậc hai | X |  |  |  |
| A2: Phương trình vô nghiệm |  | X |  |  |
| A3: Phương trình có một nghiệm kép |  |  | X |  |
| A4: Phương trình có hai nghiệm phân biệt |  |  |  | X |

***Câu 11: Xây dựng bảng quyết định cho hàm Triangle. Kiểm tra các trường hợp của tam giác.***

* Phân tích bài toán: Giả sử a, b, c là độ dài ba cạnh của tam giác. Khi đó:
  + Các biểu thức điều kiện:
    - a < b + c
    - b < a + c
    - c < a + b
    - a = b
    - a = c
    - b = c
  + Các hành động:
    - Không là tam giác
    - Tam giác thường
    - Tam giác cân
    - Tam giác đều
    - Không hợp lệ
* Bảng quyết định cho bài toán Triangle



* Nếu các điều kiện chỉ là T và F ta có 2^n cột quy tắc. Mỗi một giá trị “-” sẽ đại diện cho hai cột. Tổng số luật bằng 2^n tức là số cột quy tắc đã đủ.
* Sử dụng bảng quyết định cho Triangle ta có 11 ca kiểm thử:
  + 3 trường hợp không là tam giác
  + 1 trường hợp tam giác thường
  + 3 trường hợp là tam giác cân
  + 1 trường hợp tam giác đều
  + 3 trường hợp không hợp lệ
* Các ca kiểm thử bằng bảng quyết định cho Triangle



**Câu 12: Xây dựng hàm kiểm tra dữ liệu hợp lệ của hồ sơ học sinh gồm các trường Họ Tên, Ngày Sinh, Email và Giới tính (Nam, Nữ). Hãy cài đặt hàm này và áp dụng kiểm thử lớp tương đương để viết các kiểm thử đơn vị bằng ngôn ngữ C.**

|  |
| --- |
| 1. **typedef** **struct** hocsinh { 2. **char** hoten[35]; 3. **char** ngaysinh[30]; 4. **char** email[50]; 5. **char** gioitinh[10]; 6. } HOCSINH; 8. HOCSINH danhsach[MAX]; 9. **int** numrecords = 0; 10. **void** nhapmoi() 11. { 12. **int** done = 0; 13. **char** hoten[35]; 14. **char** ngaysinh[30]; 15. **char** email[50]; 16. **char** gioitinh[10]; 17. **do** { 18. [printf](http://cppreference.com/stdio/printf.html)("\nHo ten (trong de thoat) : "); 19. [gets](http://cppreference.com/stdio/gets.html)(hoten); 20. **if** ([strlen](http://cppreference.com/stdstring/strlen.html)(hoten) == 0) 21. done = 1; 22. **else** 23. { 24. [strcpy](http://cppreference.com/stdstring/strcpy.html)(danhsach[numrecords].hoten, hoten); 25. [printf](http://cppreference.com/stdio/printf.html)("\nNgày sinh : "); 26. [gets](http://cppreference.com/stdio/gets.html)(danhsach[numrecords].ngaysinh); 27. [printf](http://cppreference.com/stdio/printf.html)("\nEmail : "); 28. danhsach[numrecords].email; 29. [printf](http://cppreference.com/stdio/printf.html)("\nGiới tình : "); 30. danhsach[numrecords].gioitinh; 31. numrecords++; 32. } 33. } **while** (!done); 34. } |

**Câu 13: xây dựng hàm kiểm tra dữ liệu hợp lệ của hồ sơ nhân viên gồm các trường: Mã nhân viên, Họ Tên, Ngày Sinh và Giới tính (Nam, Nữ). Hãy cài đặt hàm này và áp dụng kiểm thử lớp tương đương để viết các kiểm thử đơn vị bằng ngôn ngữ C.**

**Câu 14: cho đoạn lệnh**

**Float mu(int a, int b, int c)**

**{**

1. **Float t;**
2. **If (a = = 0)**
3. **Return 0;**
4. **Int y= 0;**
5. **If((a==b)||(c==d))**
6. **y =1;**
7. **t = 1/y;**
8. **Return t;**

**}**

**xây dựng đồ thị dòng điều kiển của nó**

**Câu 15: Cho đoạn lệnh sau:**

**int giaithua( int n)**

**{**

1. **int i, t=1;**
2. **if (n = =0)**
3. **t =1;**
4. **Else if(n > 0)**
5. **{**
6. **While (i<=n)**
7. **{**
8. **t = t \* i;**
9. **i = i + 1;**
10. **}**
11. **Return t;**

**}**

**xây dựng đồ thị dòng điều kiển của thuật toán trên**

***Câu 16: Hãy nêu các độ đo kiểm thử.***

* + - 1. Uni Test (Kiểm tra mức đơn vị):
* Là hoạt động kiểm thử thực hiện trên các hàm, lớp,... hay thành phần riêng lẻ.
* Cần hiểu biết về thiết kế chương trình và code, thực hiện bởi Lập trình viên
* Mục đích: cô lập từng thành phần của chương trình và chứng minh các bộ phận riêng lẻ chính xác về các yêu cầu chức năng

1. Intergration Test (Kiểm tra tích hợp)

* Nhằm phát hiện lỗi giao tiếp xảy ra giữa các thành phần cũng như lỗi của bản thân từng thành phần (nếu có).
* Integration Test có 2 mục tiêu chính:

+ Phát hiện lỗi giao tiếp xảy ra giữa các Unit

+ Tích hợp các Unit đơn lẻ thành các hệ thống nhỏ (subsystem) và cuối cùng là nguyên hệ thống hoàn chỉnh (system) chuẩn bị cho kiểm tra ở mức hệ thống (System Test).

* Có 4 loại kiểm tra trong Integration Test.

+ Kiểm tra cấu trúc (structure): Tương tự White Box Test.

+ Kiểm tra chức năng (functional): Tương tự Black Box Test

+ Kiểm tra hiệu năng (performance): Kiểm tra việc vận hành của hệ thống.

+ Kiểm tra khả năng chịu tải (stress): Kiểm tra các giới hạn của hệ thống.

1. System Test (Kiểm tra mức hệ thống)

* Là một mức của tiến trình kiểm thử phần mềm khi các module và tích hợp các module đã được test.
* Mục tiêu: đánh giá phần mềm có tuân thủ theo các yêu cầu đã đưa ra không
* Điểm khác nhau then chốt giữa **Integration Test** và **System Test** là System Test chú trọng các hành vi và lỗi trên toàn hệ thống, còn Integration Test chú trọng sự giao tiếp giữa các đơn thể hoặc đối tượng khi chúng làm việc cùng nhau

1. Acceptance test (kiểm tra chấp nhận):

* Kiểm thử chấp nhận là một cấp độ trong tiến trình kiểm thử phần mềm nhằm kiểm thử hệ thống về khả năng chấp nhận được
* Mục tiêu của kiểm thử này là để đánh giá sự tuân thủ của hệ thống với các yêu cầu nghiệp vụ và thẩm định xem đã có thể chấp nhận để bàn giao chưa
* Gồm 2 loại kiểm thử là:
* Alpha Test, người dùng kiểm thử phần mềm ngay tại nơi phát triển phần mềm, lập trình viên sẽ ghi nhận các lỗi hoặc phản hồi, và lên kế hoạch sửa chữa.
* Beta Test, phần mềm sẽ được gửi tới cho người dùng để kiểm thử ngay trong môi trường thực, lỗi hoặc phản hồi cũng sẽ gửi ngược lại cho lập trình viên để sửa chữa.

**Câu 17: Áp dụng kiểm thử cho độ đo C1 cho đoạn lệnh của hàm UCLN**

**Câu 18: Áp dụng kiểm thử cho độ đo C2 cho đoạn lệnh của hàm UCLN**

**Câu 19: Áp dụng kiểm thử cho độ đo C3 cho đoạn lệnh của hàm UCLN**

**Câu 20: cho đoạn lệnh sau:**

**Float Trungbinhcong( int a[10], int n)**

**{**

1. **int i, tong=0, dem=0;**
2. **float tb;**
3. **i =1;**
4. **While( i<=n)**
5. **If (a[i] %2 = =0)**

**{**

**6. Tong = tong + a[i];**

**7. Dem = dem +1;**

**}**

**8.tb = (float) tong/dem;**

**9. return tb;**

**}**

**Xây dựng đồ thị dòng điều khiển ứng với độ đo C1**

**Câu 22: Cho hàm được viết bằng ngôn ngữ C hàm BinSearch**

**Int Binsearch(int x, int v[], int n)**

**{**

1. **int low = 0, high, mid;**
2. **high = n - 1;**
3. **while (low <= high)**

**{**

1. **mid = (low + high)/2;**
2. **if (x < v[mid])**
3. **high = mid - 1;**
4. **else if (x > v[mid])**
5. **low = mid + 1;**
6. **else**
7. **return mid;**

**}**

1. **return -1;**

**}**

**Xây dựng đồ thị dòng điều khiển cho hàm BinSearch ứng với độ đo C1**

**Cau 23. Cho hàm được viết bằng ngôn ngữ C hàm BinSearch**

**Int Binsearch(int x, int v[], int n)**

**{**

1. **int low = 0, high, mid;**
2. **high = n - 1;**
3. **while (low <= high)**

**{**

1. **mid = (low + high)/2;**
2. **if (x < v[mid])**
3. **high = mid - 1;**
4. **else if (x > v[mid])**
5. **low = mid + 1;**
6. **else**
7. **return mid;**

**}**

1. **return -1;**

**}**

**Hãy sinh các đường đi và các ca kiểm thử ứng với độ đo C1**

**Cau 24: Cho hàm được viết bằng ngôn ngữ C hàm BinSearch**

**Int Binsearch(int x, int v[], int n)**

**{**

1. **int low = 0, high, mid;**
2. **high = n - 1;**
3. **while (low <= high)**

**{**

1. **mid = (low + high)/2;**
2. **if (x < v[mid])**
3. **high = mid - 1;**
4. **else if (x > v[mid])**
5. **low = mid + 1;**
6. **else**
7. **return mid;**

**}**

1. **return -1;**

**}**

**Hãy sinh các đường đi và các ca kiểm thử vòng lặp while**

**Câu 25. Cho đoạn lệnh kiểm tra số nguyên tố được viết bằng ngôn ngữ C**

**int LaSoNguyenTo(int n)**

**{ int i=2;**

**Do**

**{**

**if((n % i) == 0)**

**return 0;**

**i++;**

**} while(i <= n/2);**

**return 1;**

**}**

**Xây dựng đồ thị dòng điều khiển cho hàm LaSoNguyenTo ứng với độ đo C1 và C2.**

**Cau 26: Cho hàm được viết bằng ngôn ngữ C hàm BinSearch**

**Int Binsearch(int x, int v[], int n)**

**{**

1. **int low = 0, high, mid;**
2. **high = n - 1;**
3. **while (low <= high)**

**{**

1. **mid = (low + high)/2;**
2. **if (x < v[mid])**
3. **high = mid - 1;**
4. **else if (x > v[mid])**
5. **low = mid + 1;**
6. **else**
7. **return mid;**

**}**

1. **return -1;**

**}**

**Hãy sinh các đường đi và các ca kiểm thử với độ đo C1.**

**Cau 27: Cho hàm được viết bằng ngôn ngữ C hàm BinSearch**

**Int Binsearch(int x, int v[], int n)**

**{**

1. **int low = 0, high, mid;**
2. **high = n - 1;**
3. **while (low <= high)**

**{**

1. **mid = (low + high)/2;**
2. **if (x < v[mid])**
3. **high = mid - 1;**
4. **else if (x > v[mid])**
5. **low = mid + 1;**
6. **else**
7. **return mid;**

**}**

1. **return -1;**

**}**

**Hãy sinh các đường đi và các ca kiểm thử với độ đo C2.**

**Cau 28: Cho hàm được viết bằng ngôn ngữ C hàm BinSearch**

**Int Binsearch(int x, int v[], int n)**

**{**

1. **int low = 0, high, mid;**
2. **high = n - 1;**
3. **while (low <= high)**

**{**

1. **mid = (low + high)/2;**
2. **if (x < v[mid])**
3. **high = mid - 1;**
4. **else if (x > v[mid])**
5. **low = mid + 1;**
6. **else**
7. **return mid;**

**}**

1. **return -1;**

**}**

**Sinh các ca kiểm thử để kiểm thử vòng lặp do ... while.**

**Câu 29: Cho hàm tìm UCLN được viết bằng ngôn ngữ C**

**int UCLN(int m, int n)**

**{**

1. **if (m < 0) m = -m;**
2. **if (n < 0) n = -n;**
3. **if (m == 0)**
4. **return n;**
5. **if (n == 0)**
6. **return m;**
7. **while (m != n)**

**{**

1. **if(m > n)**

**9. m = m – n;**

**10. else**

**11. n = n – m;**

**}**

1. **return m;**

**}**

**• Hãy xây dựng đồ thị dòng điều khiển cho hàm UCLN ứng với độ đo C1 và C2.**

**Cau 30: Cho hàm tìm UCLN được viết bằng ngôn ngữ C**

**int UCLN(int m, int n)**

**{**

1. **if (m < 0) m = -m;**
2. **if (n < 0) n = -n;**
3. **if (m == 0)**
4. **return n;**
5. **if (n == 0)**
6. **return m;**
7. **while (m != n)**

**{**

1. **if(m > n)**

**9. m = m – n;**

**10. else**

**11. n = n – m;**

**}**

1. **return m;**

**}**

**• Hãy sinh các đường đi và các ca kiểm thử với độ đo C1.**