```
Private Sub Quick_Sort(ByVal lower%, ByVal upper%)
    If lower >= upper Then Exit Sub

    Dim low%, midl%, high%, midval%, tmp%

    low = lower
    high = upper

    midl = lower + (upper - lower) \ 2
    midval = sortarray(midl)
    Do While low <= high
        Do Until low >= upper
            'This line is changed
            If sortarray(low) <= midval Then Exit Do
            low = low + 1
        Loop

        Do Until high <= lower
            'This line is changed
            If midval <= sortarray(high) Then Exit Do
            high = high - 1
        Loop

        If low <= high Then
            If low < high Then
                tmp = sortarray(low)
                sortarray(low) = sortarray(high)
                sortarray(high) = tmp
            End If
            low = low + 1
            high = high - 1
        End If
    Loop

    If lower < high Then Quick_Sort lower, high
    If low < upper Then Quick_Sort low, upper
End Sub
```

1. Buat flowgraph ?

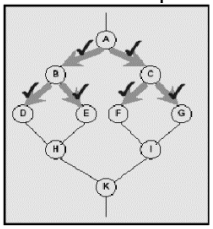
2. Hitung Cyclomatic Complexity dan tuliskan apa saja jalur-jalur independennya ?

1. Jelaskan apa yang dimaksud jalur independen ? Apa perbedaan cakupan cabang dan cakupan jalur dalam menentukan jalur independen ?

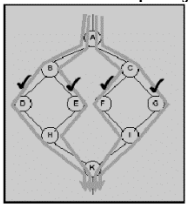
   Jawab: **Jalur independen** adalah tiap jalur pada program yang memperlihatkan 1 kelompok baru dari pernyataan proses atau kondisi baru. Untuk menentukan jumlah jalur-jalur yang independem dalam kumpulan basis suatu program dapat dilakukan dengan menghitung nilai bagi *cyclomatic complexity.*
   Cakupan cabang dan cakupan jalur adalah suatu teknik *white box testing* yang menggunakan alur logika dari program untuk membuat *test cases*. Berikut ini perbedaan antara cakupan cabang dan cakupan jalur, yaitu :

   - Cakupan cabang ditentukan dengan menilai proporsi dari cabang keputusan yang diuji oleh sekumpulan *test cases* yang telah ditentukan. Sedangkan, cakupan jalur ditentukan dengan menilai proporsi eksekusi jalur program yang diuji oleh sekumpulan *test cases* yang telah ditentukan.
   - Cakupan cabang 100% adalah bilamana tiap cabang keputusan pada program ditinjau setidaknya minimal sekali tes. Sedangkan, cakupan jalur 100 % adalah bilamana tiap jalur pada program dikunjungi setidaknya minimal sekali tes.
   - Cakupan cabang berkaitan dengan peninjauan anak panah cabang (*branch* edges) dari *flow graph*. Sedangkan, cakupan jalur berkaitan dengan peninjauan jalur sepanjang *flow graph.*

   Contoh cakupan cabang

   Contoh cakupan jalur

   Pada contoh gambar *flow graph* di samping, terdapat 6 anak panah cabang. Berdasarkan contoh *flow graph* di atas, terdapat 4 jalur.

2. Jelaskan apa persamaan dan perbedaan dari proses *debugging* yang dilakukan oleh *programmer* dan *tester* ?

   Jawab:
   Perbedaan : proses debugging merupakan proses pencarian lokasi letak bug dan menemukan apa yang salah. Programmer memperbaiki bug yang sudah diberitahu oleh tester dan membenarkan bug yang ada, tester mencari dan memberi tahu bug yang adaa tetapi tidak membenarkan bug tersebut.
   Persamaan : sama sama mencari letak kesalahan dan memperbaiki bug yang ada dan sama sama menbuat laporan terkait bug yang ada.

3. Jelaskan aspek-aspek apa saja yang harus diuji dalam configuration testing dan berikan alasannya ?

   Jawab: **Configuration testing merupakan** jenis pengujian yang bertujuan untuk memastikan bahwa perangkat lunak atau sistem dapat beroperasi dengan baik dalam berbagai konfigurasi yang mungkin digunakan oleh pengguna. Aspek-aspek yang perlu diuji dalam configuration testing melibatkan berbagai elemen konfigurasi yang dapat mempengaruhi kinerja dan kestabilan perangkat lunak. Berikut adalah beberapa aspek yang harus diuji **PC**, setiap manufaktur komputer (Dell, HP, Apple, dll) memiliki konfigurasi hardware dan software yang berbeda-beda.

   - **Components,** banyak PC dibangun oleh komponen-komponen (motherboard, card device, CD-ROM/DVD, dll) yang berbeda sesuai standar pabrikan (built up) maupun standar individu (rakitan).
   - **Peripherals,** adalah peralatan eksternal (keyboard, monitor, mouse, printer, camera, dll) yang terhubung ke PC.
   - **Interfaces,** konektor (ISA, PCI, USB, PS/2, RS/232, RJ-11, RJ-45, Fireware, dll) yang menghubungkan beberapa Komponen dan Periferal ke PC. Ada banyak kemungkinan pabrikan membuat periferal yang sama namun dengan interface yang berbeda.
   - **Options and memory,** banyak komponen dan periferal yang dijual dengan pilihan hardware dan ukuran memori yang berbeda. Misalnya: Printer bisa diupgrade agar bisa mengenali jenis font tambahan, atau kartu grafis dengan memori yang lebih besar memiliki resolusi warna yang lebih kaya, dan lain sebagainya.
   - **Device Drivers,** semua komponen dan periferal berkomunikasi dengan OS dan aplikasi lain melalui suatu driver. Driver ini umumnya disertakan dengan hardware oleh pabrikan dan akan diinstal saat hardware tersebut diinstal/diset. Meskipun driver secara teknikal termasuk dalam software namun dalam koridor testingmerupakan bagian dari konfigurasi hardware.

4. Jelaskan apa yang dimaksud dengan teknik backward compatibility testing serta uraikan hal apa saja yang menjadi pertimbangan dalam menjalankan teknik tersebut ?

   Jawab: **Backward Compatibility** adalah Suatu teknik pengujian perangkat lunak yang dilakukan untuk memastikan bahwa versi terbaru dari suatu perangkat lunak atau sistem masih dapat berinteraksi dan beroperasi dengan baik dengan versi sebelumnya. Dengan kata lain, backward compatibility testing memastikan bahwa perubahan atau pembaruan yang diterapkan pada perangkat lunak tidak mempengaruhi fungsionalitas yang ada atau ketergantungan terhadap komponen versi sebelumnya.

   Beberapa pertimbangan yang perlu diperhatikan dalam menjalankan teknik backward compatibility testing melibatkan aspek-aspek berikut:

- Fungsi yang Ada Tetap Berjalan: Pastikan bahwa semua fungsi yang ada pada versi sebelumnya masih berjalan dengan baik pada versi terbaru. Tes ini melibatkan pemeriksaan fitur-fitur kunci yang ada pada versi sebelumnya untuk memastikan bahwa mereka masih dapat digunakan tanpa kesulitan atau masalah yang signifikan.
- Antarmuka Pengguna (User Interface): Jika ada perubahan dalam antarmuka pengguna, pastikan bahwa pengguna masih dapat berinteraksi dengan perangkat lunak dengan cara yang sama atau setidaknya dengan sedikit adaptasi. Ini melibatkan pengujian navigasi, tata letak, dan fungsionalitas antarmuka pengguna.
- Kompatibilitas Data: Pastikan bahwa data yang telah ada sebelumnya masih dapat digunakan oleh versi terbaru. Ini mencakup pengujian untuk memastikan bahwa format data tidak berubah secara signifikan, dan data yang ada dapat dimuat dan disimpan dengan benar.
- Ketersediaan API (Application Programming Interface): Jika perangkat lunak melibatkan penggunaan API, pastikan bahwa API yang ada masih kompatibel dengan versi terbaru. Perubahan pada API dapat mempengaruhi integrasi dengan aplikasi lain yang mungkin menggunakan API tersebut.
- Performa: Lakukan pengujian performa untuk memastikan bahwa versi terbaru memiliki kinerja yang setidaknya setara atau lebih baik dibandingkan versi sebelumnya. Perubahan signifikan dalam kinerja dapat mempengaruhi pengalaman pengguna.
- Uji Regresi: Selain memastikan backward compatibility, juga penting untuk melakukan uji regresi secara menyeluruh untuk memastikan bahwa perubahan atau pembaruan yang diterapkan tidak merusak fungsi atau fitur lain yang sebelumnya telah diuji dan berjalan dengan baik.
- Dokumentasi: Pastikan bahwa dokumentasi yang ada telah diperbarui untuk mencerminkan perubahan pada versi terbaru. Pengguna dan pengembang harus dapat mengakses informasi terbaru mengenai perangkat lunak.

- Pembaruan Otomatis: Jika perangkat lunak mendukung pembaruan otomatis, pastikan bahwa proses pembaruan berjalan lancar dan tidak menyebabkan masalah pada instalasi yang sudah ada.

5. Jelaskan apa yang anda ketahui mengenai usability testing berdasarkan karakteristik consistent dan comfortable, dikaitkan dengan akses untuk disabilitas ?

Jawab:

Usability testing adalah suatu metode pengujian yang dirancang untuk mengevaluasi sejauh mana suatu produk atau sistem dapat digunakan dengan efektif oleh pengguna akhir. Dalam konteks aksesibilitas dan disabilitas, dua karakteristik yang penting dalam usability testing adalah consistent (konsisten) dan comfortable (nyaman).

Consistent (Konsisten): Konsistensi dalam usability testing mengacu pada keberlanjutan pola desain, interaksi, dan tata letak yang sama di seluruh produk atau sistem.
Comfortable (Nyaman): Kenyamanan dalam usability testing berarti pengguna dapat berinteraksi dengan produk atau sistem tanpa mengalami ketidaknyamanan fisik atau mental.

Ketika menggabungkan konsistensi dan kenyamanan dalam usability testing dengan fokus pada aksesibilitas, pengembang dapat menciptakan pengalaman yang lebih baik bagi pengguna dengan disabilitas. Tes khusus yang menilai keberlanjutan, kepatuhan terhadap standar aksesibilitas, dan kenyamanan bagi pengguna dengan berbagai tingkat kemampuan dapat membantu memastikan bahwa produk atau sistem dapat diakses oleh sebanyak mungkin orang.

6. Jelaskan minimal 3 perbedaan alpha testing dan beta testing dalam acceptance testing?

Jawab:

Dalam alpha testing yang dilakukan, yaitu:

- Dalam alpha testing, tester bekerjasama dengan user menguji sistem apakah sudah sesuai dengan requirement.
- Alpha testing dilakukan melalui simulasi terhadap tempat dimana sistem akan diimplementasi.
- Hal penting yang dilakukan dalam melakukan pre-delivery test
- *Alpha testing* dilakukan untuk memvalidasi produk dalam semua perspektif. Pengujian ini memastikan kesiapan produk untuk *beta testing.*
- *Alpha testing* dilakukan di akhir proses pengembangan produk.
- Pada *alpha testing*, hanya tim *developer* yang terlibat pada pengujian.
- Ketika *alpha testing* dilakukan, tim *developer* akan mendapatkan *insight* seputar *bug* atau masalah yang tidak terdeteksi ketika proses pengembangan berlangsung.

Dalam beta testing yang dilakukan, yaitu:

- Dalam beta testing, user menguji sistem apakah sudah sesuai dengan requirement di lingkungan real dan menggunakan data yang real juga.
- Beta testing dilakukan saat semua pengujian pada pre-delivery sudah dilakukan dan diterima.
- Hal tambahan yang dilakukan dalam beta testing, yaitu: Delivery check, Test Hardware, Modifikasi pre-delivery test, Pengujian terhadap semua peralatan dan Pendukung kebutuhan
- *beta testing* dilakukan untuk mendapatkan *feedback* dari pengguna. Selain itu, *beta testing* juga dilakukan untuk memastikan produk siap dirilis.
- Setelah akhir proses pengembangan produk baru dilakukan beta testing.
- Pada *beta testing* masing-masing melibatkan pengguna sebagai *tester*.
- Saat *beta testing* dijalankan, *beta tester* akan memberikan *feedback* seputar fungsi, *usability*, dan berbagai masalah lainnya yang hanya dialami oleh pengguna.

7. perbedaan take over dan acceptance ? mana dari kedua hal tersebut menurut anda yang merupakan akhir dari proses pengujian sistem dan berikan alasannya ?

Jawab:

Takeover Testing: Takeover testing, atau sering disebut juga sebagai handover testing, adalah jenis pengujian yang dilakukan ketika pengembangan sistem telah selesai dan sistem siap untuk diserahkan kepada pengguna atau tim operasional. Tujuannya adalah untuk memastikan bahwa sistem dapat dijalankan dan dikelola oleh tim yang akan mengambil alih operasionalnya.
Fokus Utama: Memastikan bahwa sistem dapat diintegrasikan dengan lingkungan produksi, dan semua konfigurasi dan pengaturan yang diperlukan sudah diterapkan.

Acceptance testing adalah jenis pengujian yang dilakukan untuk memastikan bahwa sistem atau perangkat lunak yang dikembangkan memenuhi kriteria penerimaan yang telah ditetapkan sebelumnya. Biasanya, penerimaan ini dapat dilakukan oleh pengguna akhir, klien, atau pemangku kepentingan lainnya.

Fokus Utama: Memastikan bahwa sistem memenuhi harapan pengguna dan spesifikasi yang telah ditetapkan sebelumnya. Acceptance testing dapat mencakup User Acceptance Testing (UAT), di mana pengguna akhir memvalidasi bahwa sistem memenuhi kebutuhan bisnis mereka.

Menurut saya dari kedua hal tersebut yang merupakan akhir dari proses pengujian system adalah acceptance atau penerimaan. Karena pada proses tersebut user sudah menyetujui bahwa aplikasi yang di buat oleh developer telah sesuai dengan kebutuhan yang ia inginkan dan fungsional dari system tersebut dapat berfungsi dan berjalan dengan baik. Ketika user menerima aplikasi berarti aplikasi tersebut telah sesuai dengan requirement yang diinginkan oleh user. User telah berinteraksi dengan sistem dan melakukan verifikasi apakah fungsi yang ada telah berjalan sesuai dengan kebutuhan/fungsinya. Setelah dilakukan sistem testing, acceptance testing menyatakan bahwa sistem perangkat lunak memenuhi persyaratan. Sehingga perangkat lunak bisa diterima

RANGKUMAN

PPT4.1

White box Adalah testing yang diturunkan dari "pengetahuan" tentang struktur dan implementasi program

Untuk pengujian yang lengkap maka suatu perangkat lunak harus diuji dengan white box dan black box testing

Kegunaan

- ❑ Menguji setiap jalur independent
- ❑ Menguji keputusan logic (true atau falsa)
- ❑ Menguji Loops dan batasannya
- ❑ Menguji Data Struktur internalnya

Formal review adalah kegiatan diskusi/rapat dalam static white-box testing untuk mempelajari dan menguji desain, alur logika, dan kode program, yang dilakukan oleh tester dibantu oleh programmer.

- ❑ Komponen Flow graph :
  - ➢ ***Nodes*** (titik)→ pernyataan (atau sub program) yang akan ditinjau saat eksekusi program.
  - ➢ ***Edges*** (anak panah) → jalur alur logika program untuk menghubungkan satu pernyataan (atau sub program) dengan yang lainnya.
  - ➢ ***Branch nodes*** (titik cabang) → titik-titik yang mempunyai lebih dari satu anak panah keluaran.
  - ➢ ***Branch edges*** (anak panah cabang) → anak panah yang keluar dari suatu titik cabang
  - ➢ ***Paths*** (jalur) → jalur yang mungkin untuk bergerak dari satu titik ke lainnya sejalan dengan keberadaan arah anak panah.

PPT4.2

cyclomatic complexity: pengukuran kuantitatif dari kompleksitas logika program.

Dynamic whitebox testing, adalah uji yang dilakukan dengan mengeksekusi kode program sesuai dengan testcase

Ada 4 area pengujian yang biasanya dilakukan dalam dynamic whitebox, yaitu:

- ⏵ Mengeksekusi fungsi, prosedur, subrutin, atau library dari suatu software
- ⏵ Mengeksekusi software secara utuh, dan fokus kepada operasi software yang utama
- ⏵ Mengeksekusi software di kondisi/lingkungan berbeda sesuai testcase
- ⏵ Menghitung baris kode yang dibutuhkan dalam setiap eksekusi berdasarkan testcase → efisiensi dan kecepatan

Tujuan dari dynamic whitebox testing adalah untuk menemukan bug, sedangkan tujuan dari debugging adalah untuk memperbaiki bug

PPT5

Integration testing (kadang disingkat I&T) adalah suatu tahapan proses pengujian aplikasi setelah fase unit testing dan sebelum system testing.

BIG BANG INTEGRATION

- ❑ Keuntungan

cocok untuk pengembangan aplikasi kecil/tidak kompleks

- ❑ Kerugian

Integration testing bisa dimulai ketika semua modul telah siap

lokalisasi kesalahan sulit dilakukan

mudah melewatkan kesalahan pada interface

TOP DOWN INTEGRATION

- ❑ Keuntungan
  - ▪ Lokalisasi kesalahan mudah dilakukan
  - ▪ mampu sebagai prototype awal
  - ▪ Mampu menguji modul yang telah siap terlebih dahulu
  - ▪ Kesalahan major bisa ditemukan di awal (pada level teratas)
- ❑ Kerugian
  - ▪ Kemungkinan modul level rendah tidak di tes terlalu akurat

BOTTOM UP INTEGRATION

- ❑ Keuntungan:
  - ▪ lokalisasi kesalahan lebih mudah
  - ▪ Modul dites lebih teliti
  - ▪ Testing dapat dilakukan dalam proses implementasi
- ❑ Kerugian:
  - ▪ modul tertinggi dites paling akhir
  - ▪ Kerangka aplikasi belum bisa dilihat

Configuration testing adalah proses pengujian operasi software apakah mampu berjalan/bekerja secara "normal" terhadap berbagai macam tipe hardware

FAKTA KONFIGURASI

- ❑ Kegiatan pengujian konfigurasi suatu aplikasi adalah suatu pekerjaan yang besar
- ❑ Solusinya adalah dengan membuat testcase menggunakan prinsip Equivalence Partitioning

BUG UMUM KONFIGURASI

- ❑ Dapat terjadi Bug saat aplikasi diuji pada konfigurasi yang lain
- ❑ Dapat terjadi Bug saat aplikasi diuji pada konfigurasi yang lebih spesifik/level lebih tinggi
- ❑ Dapat terjadi Bug saat aplikasi akan diubah (update/upgrade)
- ❑ Dapat terjadi Bug saat spesifikasi (mode/option) device diubah
- ❑ Dapat terjadi Bug karena kondisi tertentu dari device manufacture (waktu pakai, dll)

Langkah membuat testcase dalam configuration Testing:

1. Identifikasi komponen Input dan Output yang akan digunakan dalam suatu aplikasi
2. Definisikan fungsi, model, spesifikasi, mode, option, standar yang berlaku, dan lain sebagainya yang diperlukan dari komponen tersebut
3. Gunakan prinsip equivalence partitioning dalam pemilihan komponen (perhatikan lingkungan aplikasi)
4. Uji apakah komponen tersebut pass/fail

PPT6

Software compatibility testing yaitu menguji bagaimana interaksi antara aplikasi/software yang akan diuji tersebut dengan software lain.

Forward Compatibility adalah menguji kompatibilitas suatu aplikasi baru terhadap lingkungan platform/aplikasi lain dengan versi terbaru

LANGKAH MEMBUAT TESTCASE

1. Definisikan standar/guideline dari platform atau aplikasi lain yang harus ada di aplikasi yang akan dikembangkan
2. Buat partition equivalence (perhatikan kebutuhan kostumer dan lingkungan aplikasi)
3. Perhatikan spesifikasi produk
4. Uji kompatibilitas setiap fitur/fungsi pada aplikasi dengan memperhatikan konfigurasi hardware
5. Jika ada masalah maka justifikasi apakah major atau minor (dengan memperhatikan functional requirements)

LANGKAH USABILITY TESTCASE

- ❑ Tentukan partisipan yang terlibat dalam pengujian
- ❑ Definisikan fitur/fungsi yang akan diuji
- ❑ Definisikan ukuran setiap kriteria yang diuji
  - ▪ Jika terdapat standar ukuran atau permintaan khusus maka wajib diikuti
  - ▪ Jika tida terdapat standar ukuran maka buat baru dengan mengadakan penelitian lebih lanjut
- ❑ Kriteria merupakan tahapan interaksi yang terjadi antara user dengan sistem
- ❑ pada setiap kriteria fitur/fungsi yang diuji perhatikan aspek Follow standard/guidelines, Intuitive, Consistent, Comfortable, Correct, dan Useful

PPT7.1

- ❑ Acceptence test: Bertujuan untuk menguji apakah sistem sudah sesuai dengan apa yang tertuang dalam spesifikasi fungsional sistem (validation)
- Tipe Acceptance Testing:
- ➢ Alpha Testing, Tester bekerjasama dengan user menguji sistem apakah sudah sesuai dengan requirement
- ➢ Beta Testing, User menguji sistem apakah sudah sesuai dengan requirement di lingkungan real dan menggunakan data yang real juga

Untuk meyakinkan bahwa sistem sudah memiliki:

- ➢ Keamanan dan keselamatan dalam operasional
- ➢ Kehandalan
- ➢ Dapat dirawat secara cost-effective
- ➢ Dapat dimodifikasi untuk menyesuaikan dengan perubahan kebutuhan operasional

Test plan : dokumentasi yang menjelaskan jadwal untuk pre-delivery dan site acceptance test

Alpha testing : Dilakukan melalui simulasi terhadap tempat dimana sistem akan diimplementasi.

Beta testing : Semua pengujian pada pre-delivery sudah dilakukan dan diterima.

- ❑ Aspek lain yang harus diperhatikan:
  - ▪ Training staf yang akan mengoperasikan sistem
  - ▪ Training staf yang akan merawat sistem
  - ▪ Kebutuhan lainnya untuk tuning system, misal: max throughput, max. efisiensi, minimum cost
  - ▪ Pembuktian lainnya seperti sistem alarm, keselematan, keamanan, dan back-up
- ❑ Pengambil alihan (takeover) adalah user setuju bahwa peralatan sudah sesuai dengan kebutuhan yang ditambahkan dengan garansi untuk periode tertentu terbebas dari kesalahan
- ❑ Penerimaan (acceptance) adalah user setuju bahwa software/aplikasi sudah sesuai dengan kebutuhan