RAPPORT de TESTS d'un programme

La COMPETENCE **VERIFICATION** (UE1.2), qui consiste à **Tester un programme**, doit être travaillée pendant le semestre S1. Un des outils pour acquérir cette compétence est le **RAPPORT de TESTS**.

La COMPETENCE **VERIFICATION** se décline en trois parties :

- C2-N1-AC1 : Appliquer une procédure d'essai
- C2-N1-AC2 : Identifier un dysfonctionnement
- C2-N1-AC3 : Décrire les effets d'un dysfonctionnement

Le rapport de tests est un rapport d'exécution de votre programme, que vous faites dans le code source, sur une feuille, sous Word, sous Excel, ou autre… Chaque rapport de test est spécifique à un cahier des charges et à la façon d'analyser de chaque étudiant.

L'objectif des tests, reportés dans votre rapport, est de détecter tous les BUGS (erreur d'exécution ou non-respect du cahier des charges) et d'indiquer précisément dans quel état se trouve votre programme :

- Dans quels cas, votre programme fonctionne correctement ?
- Reste-t-il des bugs ? Où sont-ils ? A quoi sont dus ces bugs restant ?
- Comment corriger ces erreurs ? Que faudrait-il modifier dans le programme pour que les bugs disparaissent ?
- Certaines choses peuvent-elles être améliorées, même si elles ne correspondent pas, à proprement parler, à des bugs ?

Il s'agit de faire exécuter le programme avec différents jeux de tests (différentes valeurs) et de vérifier si la réponse du programme est correcte et conforme au cahier des charges. Au fur et à mesure, il faut noter les cas testés et les réponses du programmes en reportant bien les valeurs utilisées (rapport d'exécution).

Un rapport de tests est réussi lorsqu'il traduit exactement l'état de fonctionnement du programme. Un rapport de tests avec des bugs détectés peut mériter un 20/20 ; il suffit que les dysfonctionnements détectés le soient correctement et qu'ils soient analysés. Par contre un programme bugué avec un rapport de test indiquant que tout va bien conduit à une mauvaise note pour le rapport de tests.

Pour faire un bon rapport de tests, il faut reporter exactement ce que fait le programme, ne pas oublier des cas à tester et analyser ce qui se passe. Si vous détecter un bug durant la réalisation du rapport de tests et que vous le corriger, il faut mettre à jour le rapport de tests. Il doit traduire l'état du programme que récupère l'enseignant.

Voici les étapes pour réaliser un rapport de tests (reliées avec les Compétences BUT) :

- C2-N1-AC1 : Appliquer une procédure d'essai

1. Tout d'abord, **Lire le cahier des charges** et **Identifier les fonctionnalités à tester**. Chacun découpe à sa façon les points à vérifier.

2. Ensuite, **pour chaque fonctionnalité**, il faut déterminer **tous les cas à tester**, de façon à vérifier toutes les situations qui pourraient conduire à un BUG. **Un cas à tester est un ensemble de valeurs des Entrées du programme** (valeurs que doit fournir l'utilisateur lorsque le programme le lui demande).

- C2-N1-AC2 : Identifier un dysfonctionnement

3. A ce moment, le programmeur peut **lancer son programme pour chacun des cas**. Il **note**, alors, à chaque exécution**, ce que le programme affiche comme résultat** (valeur, message) ou ce qu'il fait (il va faire l'addition ou recommence l'affichage du menu…).

4. Enfin, le programmeur est arrivé à l'étape d'analyse du comportement du programme, de détection de bugs. Il **conclut pour chaque exécution précédente si le programme s'est comporté comme attendu** ou pas (OK ou BUG décelé). Pour cela, il vérifie ce qui aurait dû être produit avec la calculatrice ou la réflexion.

- C2-N1-AC3 : Décrire les effets d'un dysfonctionnement

5. Finalement, à travers une conclusion, **le testeur conclut sur les dysfonctionnements** : il les explique, montre en quoi ils posent problème.
6. **Des solutions pour résoudre les bugs ou améliorer le fonctionnement du programme** peuvent-être proposées dans cette dernière étape.

Voici des exemples de rapport de tests :

EXEMPLE Rapport de Tests du PROGRAMME 1
Calcul de la résistance équivalente à 3 résistances montées en parallèle

A) Fonctionnalité à tester : SAISIES DES RESISTANCES DU MONTAGE

CAS TESTES :	E (Entrées du programme) R1 R2 R3 (Ohms)	RESULTAT du programme	BILAN
erreur saisie 1 (nul)	**0 0 0**	req=0 Ohm	**BUG**
erreur saisie 2 (nul)	**0 1 1**	req=0 Ohm	**BUG**

erreur saisie 3 (nul)	**1 0 1**	req=0 Ohm	**BUG**
erreur saisie 4 (nul)	**1 1 0**	req=0 Ohm	**BUG**
erreur saisie 5 (nul)	**0 0 1**	req=0 Ohm	**BUG**
erreur saisie 6 (nul)	**0 1 0**	req=0 Ohm	**BUG**
erreur saisie 7 (nul)	**1 0 0**	req=0 Ohm	**BUG**
erreur saisie 8 (négatif)	**-1 1 1**	req=1.00 Ohm	**BUG**
erreur saisie 9 (négatif)	**1 -1 1**	req= 1.00 Ohm	**BUG**
erreur saisie 10 (négatif)	**1 1 -1**	req= 1.00 Ohm	**BUG**
erreur saisie 11 (négatif)	**-1 -1 -1**	req=--0.33 Ohm	**BUG**
erreur saisie 12 (négatif)	**-1 -1 1**	req=--1.00 Ohm	**BUG**
erreur saisie 13 (négatif)	**-1 1 -1**	req=--1.00 Ohm	**BUG**
erreur saisie 14 (négatif)	**1 -1 -1**	req=--1.00 Ohm	**BUG**

Conclusion :

Lorsqu'au moins une des résistances est nulle (présence de court-circuit dans le schéma électrique), le programme présente un BUG : il calcule 0 comme résistance équivalente. Ceci est faux car la division par 0 est impossible, donc il ne doit pas y avoir de résultat dans ce cas là. En fait, le calcul fait une double inversion de 0, ce qui "cache" les divisions par 0.(0->infini->0).

Lorsqu'au moins une des résistances est négative (impossible dans un circuit électrique), le programme présente un BUG : il calcule quand même une résistance équivalente.

Les erreurs de saisie conduisant à des calculs impossibles (ri=0) ou à des valeurs impossibles physiquement (ri<0) ne sont pas gérées par le programme.

La solution à ces bugs consisterait à tester les résistances et si au moins une résistance est nulle ou négative, un message d'erreur doit être affiché pour expliquer son erreur à l'utilisateur ; dans le cas contraire (aucune résistance nulle ou négative), le calcul de la résistance équivalente peut être effectué. Il faudrait donc ajouter une alternative ou, mieux, une boucle pour gérer les erreurs de saisie de l'utilisateur ; la boucle donnerait une chance à l'utilisateur de refaire ses saisies correctement, voir, même l'obligerait à entrer des valeurs acceptables. De plus, des messages clairs et précis sur les demandes aideraient l'utilisateur du programme à ne pas se tromper.

B) Fonctionnalité à tester : CALCULS RESISTANCE EQUIVALENTE DU MONTAGE PARALLELE

	E (Entrées du programme) R1 R2 R3 (Ohms)	RESULTAT du programme	BILAN
Cas 1	**1 1 1**	req=0.33 Ohm	**OK**
Cas 2	**0.5 4.5 10**	req=0.43 Ohm	**OK**
Cas 3	**10 20 100**	req=6.25 Ohm	**OK**

Conclusion :

Dans le cas de valeurs de résistances ne produisant pas d'erreur de calcul ou d'aberration physique, les valeurs des résistances équivalentes parallèles sont exacts (vérifiés à la calculatrice).

EXEMPLE Rapport de Tests du PROGRAMME 2
Calcul de la vitesse d'un marathonien

A) Fonctionnalité à tester : SAISIES DES PERFORMANCES D'UN MARATHONIEN

CAS TESTES :	E (Entrées du programme) dist (Km)	h	mn	s	RESULTAT du programme	BILAN
3 paramètres de temps nul	**10**	**0**	**0**	**0**	v= 1.#J m/s	**BUG**
Distance négative	**-10**	**1**	**0**	**0**	v= -2.78 m/s	**BUG**
Temps négatif	**10**	**-1**	**0**	**0**	v= -2.78 m/s	**BUG**

Conclusion :

Lorsque les 3 paramètres de temps sont nuls (le coureur est instantanément arrivé), le programme présente un BUG : il affiche des caractères comme valeur de vitesse. En effet, la division par 0 étant impossible, le calcul plante.
Lorsque la distance est négative (impossible physiquement), le programme présente un BUG : il calcule quand même une vitesse qui s'avère être négative (impossible).
Lorsque la durée découlant des nombres d'heures, de minutes et de secondes est négatif (impossible physiquement), la vitesse calculée est négative, ce qui n'est pas réaliste.
Les erreurs de saisie conduisant à des calculs impossibles (diviseur nul) ou à des valeurs impossibles physiquement (distance ou temps négatif) ne sont pas gérées par le programme.

La solution à ces bugs consisterait à tester les paramètres de temps et s'ils sont tous les 3 nuls ou si l'un au moins est négatif, un message d'erreur doit être affiché pour expliquer son erreur à l'utilisateur ; dans le cas contraire (heure, minute et seconde non tous nuls en même temps et aucune de ces valeurs n'est négative), le calcul de la vitesse de course peut être effectué. D'autre part, le programme doit également empêcher que la valeur de distance soit négative, car c'est impossible.
Il faudrait donc ajouter des alternatives ou, mieux, des boucles pour gérer les erreurs de saisie de l'utilisateur ; la boucle donnerait une chance à l'utilisateur de refaire ses saisies correctement, voir, même l'obligerait à entrer des valeurs acceptables. De plus, des messages clairs et précis sur les demandes aideraient l'utilisateur du programme à ne pas se tromper.

B) Fonctionnalité à tester : CALCUL DE LA VITESSE D'UN MARATHONIEN

CAS TESTES :	E (Entrées du programme) dist (Km)	h	mn	s	RESULTAT du programme	BILAN
Cas 1 : Spyridon Louis 1896	**42**	**2**	**58**	**50**	v= 3.91 m/s	**OK**
Cas 2 : Makau Patrick 2011	**42**	**2**	**3**	**38**	v= 5.66 m/s	**OK**

Conclusion :

Dans le cas de valeurs de temps et de distance ne produisant pas d'erreur de calcul ou d'aberration physique, les valeurs des vitesses de course sont exactes (vérifiés sur le WEB). Nous pouvons également noter que les performances des athlètes ont très fortement progressées en un siècle !

EXEMPLE Rapport de Tests du PROGRAMME 3
Calculatrice arithmétique à 4 opérations

A) Fonctionnalité à tester : SAISIE OPERATION DANS LE MENU

CAS TESTES :	E (Entrées du programme) Choix menu Opération	RESULTAT du programme	BILAN
Cas OPTION 1	**1**	Vers addition et saisie opérandes	**OK**
Cas OPTION 2	**2**	Vers soustraction et saisie opérandes	**OK**
Cas OPTION 3	**3**	Vers multiplication et saisie opérandes	**OK**
Cas OPTION 4	**4**	Vers division et saisie opérandes	**OK**
Cas OPTION 5	**5**	Sort du programme	**OK**
Cas erreur 1	**0**	« ERREUR CHOIX MENU » puis menu	**OK**
Cas erreur 2	**6**	« ERREUR CHOIX MENU » puis menu	**OK**

Conclusion :

Le choix d'une option valide d'opération (choix de 1 à 4) dirige bien l'exécution du programme vers la bonne opération mathématique, avec saisie préalable des opérandes pour le calcul.
Le choix de l'option 5 fait bien arrêter la calculatrice, ce qui est conforme au fait que c'est l'option de Sortie.
Une erreur de choix Menu (valeurs strictement inférieures à 1 et strictement supérieures à 5), donc en particulier avec les valeurs limites 0 et 6, conduit à l'affichage d'un message d'erreur avant que le menu soit reproposé à l'utilisateur. Les erreurs de choix menu sont donc bien gérées.

Améliorations il serait possible d'être plus précis dans le message d'erreur utilisateur, en lui rappelant que les choix autorisés sont compris entre 1 et 5.

B) Fonctionnalité à tester : SAISIE OPERANDES DE CALCUL

CAS TESTES :	E (Entrées du programme) Choix menu	a	b	RESULTAT du programme	BILAN
Cas 1	**1**	**1.5**	**0**	addition effectuée	**OK**
Cas 2	**2**	**1.5**	**0**	soustraction effectuée	**OK**
Cas 3	**3**	**1.5**	**0**	multiplication effectuée	**OK**
Cas 4	**4**	**1.5**	**2**	division effectuée	**OK**
Cas 5	**4**	**1.5**	**0**	« diviseur nul ! » puis menu	**OK**

Conclusion :

Le choix des calculs arithmétiques 1 à 3 conduit à la réalisation du calcul demandé, quel que soit les valeurs des opérandes proposées par l'utilisateur du programme. Il n'existe, dans ces cas, aucune impossibilité de calcul.

Par contre, dans le cas de la division, si l'utilisateur choisit un diviseur nul, un message d'erreur s'affiche et le menu général de choix d'opération est réaffiché.

Les erreurs de calcul sont partiellement gérées.

Améliorations il serait possible d'éviter à l'utilisateur de revenir au menu général en cas d'erreur de saisie du diviseur, mais de revenir directement à une nouvelle division. Pour cela, il faut utiliser une boucle qui recommence la saisie de b tant que b est nul ; l'utilisateur est alors forcé à entrer une valeur de diviseur valide. D'autre part des messages de demande plus précis pourrait permettre d'anticiper sur d'éventuelles erreurs de l'utilisateur en lui expliquant clairement ce qui est attendu de lui.

C) Fonctionnalité à tester : CALCULS ARITHMETIQUES

CAS TESTES :	E (Entrées du programme) Choix menu a b	RESULTAT du programme	BILAN
Cas 1 (+)	**1** **5** **2**	resu= 7	**OK**
Cas 2 (-)	**1** **5** **2**	resu= 3	**OK**
Cas 3 (x)	**1** **5** **2**	resu= 10	**OK**
Cas 4 (/)	**1** **5** **2**	resu= 2.5	**OK**

Conclusion :

Les résultats des calculs sont corrects lorsque les calculs sont possibles.