Workshop tentamen Advanced C Programming D-B-ESOT-AC

***Toelichting op dit tentamen***

Bij dit tentamen wordt de theoretische en praktische kennis van onderwerpen behorende bij de OWE Advanced C Programming getoetst.

Toetsing vindt plaats door het binnen de beschikbare tentamentijd van 3 lesuren drie compleet uitgewerkte opdrachten te maken. In ieder van deze programma's behandelen we een deel van de stof die behoort tot de leeruitkomsten van Advanced C Programming

Bij dit tentamen mag je gebruik maken van:

* Het boek Advanced C programming
* Eventuele aantekeningen
* Een laptop met het beschikbaar gestelde Ubuntu 20.x image

***Gedurende het tentamen mag je geen verbinding maken met het internet, dus tijdens je tentamen staan de netwerkverbindingen uit.***

***Dit is een individueel tentamen, je mag gedurende het tentamen op geen enkele wijze communiceren met anderen (zowel studenten als buitenstaanders).***

Je ontwerpt, ontwikkelt en realiseert de gevraagde programma functionaliteit en onderdelen.   
Opdrachten worden uitgewerkt onder Linux op de in de les gebruikte Ubuntu 20.x image

* Als er om functies wordt gevraagd worden deze correct en werkend gecodeerd.
* Er wordt gecodeerd volgens de geldende code-style
* Alle code is compileerbaar er zijn geen errors of warnings
* De functie wordt opgenomen in de daarvoor bestemde (of te maken) module.
* Als je een deel van een opdracht niet kunt maken, gebruik dan een zelf ingestelde waarde of functie om toch met de vervolgopdrachten verder te gaan.  
  Voorbeelden:
  1. Als opdracht 1a niet lukt, kies dan bij opdracht 1b zelf een waarde voor *count*
  2. Als het niet lukt om de opdrachten die met een SLL te maken hebben te maken, zorg dan dat je wel de commandline parameters opdrachten verwerkt en toon aan dat je dat onderdeel wel kunt toepassen.

**Inleveren:**

Maak een zip van de complete directory met opgaven en lever deze in via HANDIN. Dit kan door gebruik te maken van *firefox* en het adres handin.han.nl. Je kunt het inleveren bij:

D-B-ESOT-AC Tentamen Advanced C programmeren

De zipfile maak als volgt:

cp -r ADC-tentamen ADC-tentamen.save

zip -r ADC-tentamen.zip ADC-tentamen

(let op zip op linux heeft eerst de destination en dan hetgeen je wilt "zippen")

***Beoordeling:***

|  |  |  |
| --- | --- | --- |
| ***Opdracht 1*** | ***Over de Singly Linked List*** | ***35*** |
| ***Opdracht 2*** | ***Unit testen met Unity*** | ***25*** |
| ***Opdracht 3*** | ***Dynamic Memory management*** | ***20*** |
| ***Opdracht 4*** | ***Function pointers*** | ***20*** |
| ***Cesuur (5.5)*** |  | ***55*** |
|  |  |  |

***Opdracht 1, Singly Linked List en Commandline parameters 35 punten***

*Deze opdracht bestaat uit 3 delen a t/m c.*

Maak voor deze code gebruik van het raamwerk dat beschikbaar is gesteld onder de naam:

***ADC-2021p1-opdracht1***

Bestudeer de code nauwkeurig, houd er rekening mee dat de code niet exact hetzelfde is als bij de vergelijkbare voorbeelden in het boek.

De code in het raamwerk wordt gebruikt om de dynamische structuur Singly Linked List (SLL) te onderhouden.

|  |  |
| --- | --- |
| Opdracht | Punten |
| 1. *Aanvullen van de raamwerk code*   Vul de code in int main(int argc, char \*argv[]) aan zodat het programma als volgt kan worden aangeroepen:  $ ./sll 5   * Zorg dat de executable van het programma sll heet, gebruik make of  gcc main.c sll.c -o sll * Controleer of het juiste argument aan het commando wordt meegegeven (2 punten). * Zet de waarde van het argument in de variabele count (7 punten) * Druk count af op het scherm (1 punt)   *tip: gebruik int atoi(const char \*str) om een string om te zetten in een integer.* | 10 |
| 1. *Dynamische uitbreiden van de SLL*   Pas de int main(int argc, char \*argv[])functie aan zodat er een SLL gemaakt wordt met count nodes. (10 punten)  De waarde van data loopt mee met het nummer van de node  *Let op! Alleen als het niet gelukt is om opdracht a. goed uit te voeren, laat hier dan op count een initiële waarde staan:*  *int count = 4;* | 10 |

|  |  |
| --- | --- |
| 1. *Functiebibliotheek voor SLL uitbreiden*  Maak een functie met de volgende functionaliteit:   De functie moet een node toevoegen op elke willekeurige plaats in de SLL.  De functie heeft als prototype:  *void insertSLL(node\_t \*\*pHead, int data, unsigned int node);*  Er wordt een nieuwe node toegevoegd ***na*** node nummer *node*  Er hoeft niet gecontroleerd te worden of node groter is dan het aantal nodes in de SLL.  ***Laat in main.c zien dat de functie werkt***  *TIP: Maak eerst voor jezelf een tekening zoals die gemaakt kan worden voor SLL en bepaal dan wat er moet gebeuren om een node in een SLL toe te voegen.* | 15 |

***Opdracht 2, Unity gebruiken voor het testen van een functie 25 punten***

*Deze opdracht bestaat uit 3 delen a t/m c*

Maak bij deze opdracht gebruik van de raamwerk code zoals deze beschikbaar is gesteld:

**ADC-2122p1-opdracht2**

In de opdracht is een raamwerk opgenomen om met gebruik van het Unity framework functies te testen.

De functie die getest wordt:  
  
 *int compareDoubles(const double D1, const double D2, epsilon);*

De functie is opgenomen in functions.h/functions.c, in het commentaar wordt de functie toegelicht.

|  |  |
| --- | --- |
| 1. Maak een testfunctie void test\_compareDoublesEqual(void);   Voer daarin minimaal 4 tests uit die de functie compareDoubles() test op D1 is ***gelijk*** aan D2. Maak gebruik van een *epsilon* van *0.0002*. Maak bij het testen gebruik van de mogelijkheden van het Unity framework, kies een geschikte TEST\_.....  Zorg dat er ook een *boundary test*s worden uitgevoerd.  Geef in het commentaar aan waarom je bepaalde testwaarden hebt gekozen. | **10** |
| 1. Maak een testfunctie void test\_compareDoubles**Not**Equal(void);  Voer daarin minimaal 4 tests uit die de functie compareDoubles() test op D1 ***groter of kleiner*** D2. Maak gebruik van een *epsilon* van *0.0002*. Maak bij het testen gebruik van de mogelijkheden van het Unity framework, kies een geschikte TEST\_.....   Zorg dat er ook een *boundary test*s worden uitgevoerd.  Geef in het commentaar aan waarom je bepaalde testwaarden hebt gekozen. | **10** |
| 1. Maak de *int main(void)* functie compleet zodat de test functies, passend binnen het Unity framework worden uitgevoerd.  Voer het programma uit, zorg dat de output in een bestand komt te staan door gebruik te maken van:  ./functions-test > out.txt | **5** |

**Opmerkingen/tips:**

1. *er staat een makefile in deze directory zodat je door gebruik te maken van make het geheel kan compileren en linken. Resultaat is functions-test die je met ./functions-test kunt uitvoeren*
2. *Gebruik je geen make dan moet je het geheel compileren en linken met:  
     
   gcc main.c functions.c Unity/unity.c -o functions-test*
3. *Voor de volledigheid is de Unity documentatie opgenomen, pas op dat je niet alles door gaat nemen, maar je kunt wel kijken welke TEST\_ASSERT\_..... 's er beschikbaar zijn.*
4. *Er zitten een paar warnings in de Unity versie, deze kun je negeren.*

**Opdracht 3, Controleren van een programma op correct coderen en geheugengebruik (20 punten)**

Maak bij deze opdracht gebruik van de raamwerk code zoals deze beschikbaar is gesteld voor

**ADC-2122p1-opdracht3**

het programma valgrind biedt de mogelijkheid om onder andere op memory leaks te controleren. De volgende opdrachten hebben betrekking op het alloceren en vrij geven van geheugen. Controleer met valgrind of er een memory leak in het programma zit. Zoek het memory leak op, vermeldt het in het commentaar en probeer het memory leak te repareren.

|  |  |
| --- | --- |
| 1. ***Memory leak bij gebruik van SLL***   De code staat in subdirectory   **ADC-2122p1-opdracht3-base/sll** Compileer het programma met make of met:  gcc main.c sll-faulty.c -o sll  controleer het programma met:  valgrind ./sll  of  valgrind --tool=memcheck –leak-check=yes ./sll  Is er een memory leak?  Repareer de memory leak als deze er is. | **10** |
| 1. ***Memory leak in het programma report\_user***   De code staat in subdirectory  **ADC-2122p1-opdracht3-base/report\_user**  Compileer het programma met make of met:  gcc main.c report\_user.c -o report\_user  controleer het programma met:  valgrind ./report\_user  of  valgrind --tool=memcheck –leak-check=yes ./sll  Is er een memory leak?  Repareer de memory leak als deze er is. | **10** |

**Opdracht 4, Een calculator met functie pointers (20 punten)**

Maak bij deze opdracht gebruik van de raamwerk code zoals deze beschikbaar is gesteld voor

**ADC-2122p1-opdracht4**

In deze code is een eenvoudige rekenmachine geïmplementeerd die gebruik maakt function pointers/callback functies.

|  |  |
| --- | --- |
| Bestudeer de code en vul de code aan zodanig dat de array operations is gevuld met de juiste informatie om de 4 operaties (sum, sub, mult en divd) middels functie pointers uit te kunnen voeren:   * Maak de typedef van de struct compleet zodat er een functie pointer met naam ope in staat. * Zorg dat de juiste functiepointers in de array operations komen te staan | **20** |