Zadanie 2 – Binárne rozhodovacie diagramy

Vytvorte program, v ktorom bude možné vytvoriť dátovú štruktúru BDD (Binárny Rozhodovací Diagram) so zameraním na využitie pre reprezentáciu Booleovských funkcií.

Konkrétne implementujte tieto funkcie:

- BDD *BDD_create (string *bfunkcia,* string *poradie*);
- BDD *BDD_create_with_best_order (string *bfunkcia*);
- char BDD_use (BDD **bdd*, string *vstupy*);

Samozrejme môžete implementovať aj ďalšie funkcie, ktoré Vám budú nejakým spôsobom pomáhať v implementácii vyššie spomenutých funkcii, nesmiete však použiť existujúce funkcie na prácu s binárnymi rozhodovacími diagramami.

Funkcia **BDD_create** má slúžiť na zostavenie redukovaného binárneho rozhodovacieho diagramu, ktorý má reprezentovať/opisovať zadanú Booleovskú funkciu, ktorá je zadaná ako argument funkcie **BDD_create** (argument *bfunkcia*). Booleovská funkcia je poskytnutá funkcii **BDD_create** v tvare výrazu (presný formát a dátový typ je na Vás, napríklad DNF string). Druhým argumentom je *poradie* premenných, ktorým sa definuje, v akom poradí sú použité jednotlivé premenné Booleovskej funkcie bfunkcia na vytvorenie BDD (opäť formát a dátový typ si môžete zvoliť akýkoľvek, napr. spájaný zoznam alebo pole premenných, ktoré sú očíslované od 0 po N-1). Návratovou hodnotou funkcie **BDD_create** je ukazovateľ na zostavený (a zároveň redukovaný) binárny rozhodovací diagram (podľa zvoleného poradia premenných), ktorý je reprezentovaný vlastnou štruktúrou BDD. Štruktúra BDD musí obsahovať minimálne tieto zložky: počet premenných, veľkosť BDD (počet uzlov) a ukazovateľ na koreň (prvý uzol) BDD. Samozrejme potrebujete aj vlastnú štruktúru, ktorá bude reprezentovať jeden uzol BDD (ako vrchol v strome). Súčasťou tejto funkcie je nielen vytvorenie BDD, ale aj samotná redukcia BDD – môžete si vybrať, či redukciu vykonáte až po zostavení úplného BDD, alebo či redukujete BDD priebežne už počas jeho vytvárania. Ak sa však rozhodnete redukovať BDD až po jeho úplnom zostavení, riešenie je penalizované 4 bodmi pre nižšiu efektivitu (vyššiu časovú zložitosť).

Funkcia **BDD_create_with_best_order** má slúžiť na nájdenie čo najlepšieho poradia (v rámci vyskúšaných možností) premenných pre zadanú Booleovskú funkciu. Hľadanie spočíva v opakovanom volaní funkcie **BDD_create**, pričom sa skúšajú použiť rozličné poradia premenných (argument *poradie*). Napríklad pre Booleovskú funkciu s 5 premennými môžeme zavolať **BDD_create** 5-krát, s použitím poradia 01234, 12340, 23401, 34012 a 40123. Alebo môžeme vyskúšať aj všetky možné permutácie poradia premenných, ktorých je N!, kde N je počet premenných (t.j. v tomto prípade 5! = 120). Alebo môžeme použiť X náhodne zvolených poradí premenných. Dôležite však je, aby sa vyskúšalo aspoň N ľubovoľných unikátnych poradí premenných, kde N je počet premenných Booleovskej funkcie. Návratovou hodnotou tejto funkcie je ukazovateľ na BDD, pričom sa použije BDD s najnižším počtom uzlov zo všetkých vyskúšaných poradí premenných.

Funkcia **BDD_use** má slúžiť na použitie BDD pre zadanú (konkrétnu) kombináciu hodnôt vstupných premenných Booleovskej funkcie a zistenie výsledku Booleovskej funkcie pre túto kombináciu vstupných premenných. V rámci tejto funkcie „prejdete“ BDD stromom smerom od koreňa po list takou cestou, ktorú určuje práve zadaná kombinácia hodnôt vstupných premenných. Argumentami funkcie **BDD_use** sú ukazovateľ s názvom *bdd* ukazujúci na BDD (ktorý sa má použiť) a ukazovateľ

s názvom *vstupy* ukazujúci na začiatok reťazca (pole znakov). Práve tento reťazec / pole znakov reprezentuje nejakým (vami zvoleným) spôsobom konkrétnu kombináciu hodnôt vstupných premenných Booleovskej funkcie. Napríklad, index poľa reprezentuje nejakú premennú a hodnota na tomto indexe reprezentuje hodnotu tejto premennej (t.j. pre premenné A, B, C a D, kedy A a C sú jednotky a B a D sú nuly, môže ísť napríklad o "1010"), môžete si však zvoliť iný spôsob. Návratovou hodnotou funkcie **BDD_use** je char, ktorý reprezentuje výsledok Booleovskej funkcie – je to buď '1' alebo '0'. V prípade chyby (napr. chybný vstup) je táto návratová hodnota záporná (napr. -1).

Okrem implementácie samotných funkcií na prácu s BDD je potrebné vaše riešenie dôkladne otestovať. Vaše riešenie musí byť 100% korektné. V rámci testovania je potrebné, aby ste náhodným spôsobom generovali Booleovské funkcie, podľa ktorých budete vytvárať BDD pomocou funkcie **BDD_create** a funkcie **BDD_create_with_best_order**. Správnosť BDD môžete overiť opakovaným (iteratívnym) volaním funkcie **BDD_use** tak, že použijete postupne všetky možné kombinácie hodnôt vstupných premenných a porovnávate výsledok **BDD_use** s očakávaným výsledkom, ktorý získate vyhodnotením výrazu. Testujte a vyhodnoťte Vaše riešenie pre rozličný počet premenných Booleovskej funkcie (čím viac premenných Váš program zvládne, tým lepšie. Mal by byť aspoň 13). Počet rozličných Booleovských funkcií / BDD diagramov pre rovnaký počet premenných Booleovskej funkcie by mal byť minimálne 100. V rámci testovania tiež vyhodnocujte percentuálnu mieru zredukovania BDD (t.j. počet odstránených uzlov / počet uzlov pre úplný diagram). Taktiež vyhodnoťte dodatočnú percentuálnu mieru zredukovania BDD dosiahnutú skúšaním rôznych poradí premenných, t.j. porovnajte veľkosť BDD vytvoreného priamo funkciou **BDD_create** s veľkosťou BDD získaného funkciou **BDD_create_with_best_order**.

Príklad veľmi jednoduchého testu (len pre pochopenie problematiky):

```
#include <string.h>
int main(){
  BDD* bdd;
  bdd = BDD_create("AB+C");
  if (BDD_use("000") != '0') /* ocakavanu hodnotu vieme zistit
dosadenim hodnot vstupnych premennych do vyrazu – lepsie je spravit
funkciu na dosadenie tychto hodnot a vypocitanie vysledku */
    printf("error, for A=0, B=0, C=0 result should be 0.\n");
  if (BDD_use("001") != '1')
    printf("error, for A=0, B=0, C=1 result should be 1.\n");
  if (BDD_use("010") != '0')
    printf("error, for A=0, B=1, C=0 result should be 0.\n");
  if (BDD_use("011") != '1')
    printf("error, for A=0, B=1, C=1 result should be 1.\n");
  if (BDD_use("100") != '0')
    printf("error, for A=1, B=0, C=0 result should be 0.\n");
  if (BDD_use("101") != '1')
    printf("error, for A=1, B=0, C=1 result should be 1.\n");
  if (BDD_use("110") != '1')
    printf("error, for A=1, B=1, C=0 result should be 1.\n");
  if (BDD_use("111") != '1')
    printf("error, for A=1, B=1, C=1 result should be 1.\n");
  return 0;
}
```

Okrem implementácie vášho riešenia a jeho testovania vypracujte aj dokumentáciu, v ktorej opíšete vaše riešenie, jednotlivé funkcie, vlastné štruktúry, spôsob testovania a výsledky testovania, ktoré by

mali obsahovať (priemernú) percentuálnu mieru zredukovania BDD a (priemerný) čas vykonania vašich funkcií. Samozrejme aj v závislosti od počtu premenných Booleovskej funkcie. Dokumentácia musí obsahovať hlavičku (kto, aké zadanie odovzdáva), stručný opis použitého algoritmu s názornými nákresmi/obrázkami a krátkymi ukážkami zdrojového kódu, vyberajte len kód, na ktorý chcete extra upozorniť. Pri opise sa snažte dbať osobitý dôraz na zdôvodnenie správnosti vášho riešenia – teda dôvody prečo je dobré/správne, spôsob a vyhodnotenie testovania riešenia. Nakoniec musí technická dokumentácia obsahovať odhad výpočtovej (časovej) a priestorovej (pamäťovej) zložitosti vášho riešenia (**BDD_create** a **BDD_create_with_best_order**). Celkovo musí byť cvičiacemu jasné, že viete čo ste spravili, že viete odôvodniť, že to je správne riešenie, a viete aké je to efektívne.

Riešenie zadania sa odovzdáva do miesta odovzdania v AIS do stanoveného termínu (oneskorené odovzdanie je prípustné len vo vážnych prípadoch, ako napr. choroba, o možnosti odovzdať zadanie oneskorene rozhodne cvičiaci, príp. aj o bodovej penalizácii). Odovzdáva sa jeden **zip** archív, ktorý obsahuje zdrojové súbory s implementáciou riešenia a testovaním + jeden súbor s dokumentáciou vo formáte **pdf**. **Vyžaduje sa tiež odovzdanie programu**, ktorý slúži na testovanie a odmeranie efektívnosti týchto implementácií ako jedného samostatného zdrojového súboru (obsahuje funkciu **main**).

Hodnotenie

Môžete získať celkovo 20 bodov, **nutné minimum je 8 bodov**.

Za implementáciu riešenia je možné získať celkovo 12 bodov, z toho 8 bodov za funkciu **BDD_create**, 2 body za funkciu **BDD_create_with_best_order** a 2 body za funkciu **BDD_use**. Za testovanie je možné získať 4 bodov (z toho 2 body za automatické zistenie očakávaného výsledku BDD – dosadenie hodnôt do výrazu a jeho vyhodnotenie) a za dokumentáciu 4 body (bez funkčnej implementácie 0 bodov). Body sú ovplyvnené aj prezentáciou cvičiacemu (napr. keď neviete reagovať na otázky vzniká podozrenie, že to **nie je vaša práca, a teda je hodnotená 0 bodov)**. Ak Vaše riešenie nevykonáva žiadnu redukciu alebo ak vytvorený BDD nefunguje 100% správne, riešenie nie je akceptované.