## Esame 20250107

### Esercizio 2

(1) **Esercizio 2 v1**

ESSAY | marked out of 10 | penalty 0 | File picker

Il formato DIMACS CNF è una rappresentazione testuale di una formula in forma normale congiuntiva. Una formula in forma normale congiuntiva è una congiunzione (logica "and") di un insieme di clausole. Ogni clausola è una disgiunzione (logica "or") di un insieme di letterali. Un letterale è una variabile o una negazione di una variabile. DIMACS CNF utilizza gli **interi positivi** per rappresentare le variabili e la loro negazione (valore negativo) per rappresentare la corrispondente variabile negata. Un file DIMACS cnf contiene una intestazione (prima riga) così formata: `p cnf <numero clausole> <numero variabili>`, seguita da una serie di clausole. Ogni clausola inizia con un numero positivo che specifica quanti letterali ci sono nella clausola. Ad esempio, la formula $(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$ può essere rappresentata nel formato DIMACS CNF come segue:

```
p cnf 2 3
3 1 2 -3
3 -1 -2 3
```

Un assegnamento di valori alle variabili che rende la formula vera è detto **assegnamento soddisfacibile**. Un assegnamento consiste nell'assegnare un valore vero o falso ad ogni variabile. Ad esempio, l'assegnamento $\{x_1 = 1, x_2 = 0, x_3 = 1\}$ rende vera la formula sopra. Un assegnamento che rende la formula falsa è detto **assegnamento insoddisfacibile**. Una formula che non ha assegnamenti soddisfacibili è detta **insoddisfacibile**.

Un file DIMACS é rappresento in memoria con una struttura `Dimacs` che contiene il numero di clausole, il numero di variabili e le clausole rappresentati con una lista concatenata `cnf_t`, che a sua volta contiene una lista di letterali rapresentati con una lista concatenata `List` di interi (Si veda il file `esercizio2.cpp`).

Un assegnamento é rappresentato in memoria con una struttura `Assignment` che contiene un array di booleani tale che l'elemento `i` dell'array rappresenta il valore dell'assegnamento della variabile `i+1` (1 vero, 0 falso).

Il programma in `esercizio2.cpp` prende come argomento a) un file che rappresenta una formula in formato DIMACS e costruisce e popola la struttura `Dimacs`, b) un file che rappresenta un assegnamento parziale alle variabili della formula e costruisce e popola la struttura `Assignment`, e c) chiama una funzione `check_assigment` (da definire) che prende come primo argomento una struttura `Dimacs` e come secondo argomento una struttura `Assignment`, entrambe passate per *riferimento* e stampa a video `Assignment is satisfying the formula` se `check_assigment` ritorna `true`, altrimenti stampa `Assignment is not satisfying the formula`.

La funzione `check_assigment` ritorna `true` se e solo se l'assegnamento é un *assegnamento soddisfacibile*.

La funzione `check_assigment` **deve essere ricorsiva** e **NON deve contenere iteratori** espliciti (`for`, `while`, `do-while`). La funzione `check_assigment` può ovviamente contenere codice sequenziale o condizionale. Sono consentite (se ritenute necessarie) chiamate a funzioni ricorsive ausiliarie che a loro volta **non contengano iterazioni esplicite** (`for`, `while`, `do-while`).

Il file `esercizio2.cpp` contiene tutto quanto necessario tranne la dichiarazione e la definizione della funzione `check_assigment`. Di seguito è riportato un esempio di esecuzione.

```
computer > ./a.out input.cnf inputsat.ass
Dimacs file: input.cnf
p cnf 10 30
3 20 -10 5
2 -2 -3
4 -4 -8 9 30
3 22 -11 15
2 -21 -9
4 -14 -18 19 -30
4 -24 -28 29 30
3 12 -21 25
2 -1 -29
4 -24 -28 29 -10
Assignment file: inputsat.ass
1 -2 3 4 5 -6 -7 -8 -9 -10 -11 12 -13 -14 15 -16 -17 -18 -19 -20
 21 -22 -23 -24 -25 -26 -27 -28 -29 30
Assignment is satisfying the formula
computer > ./a.out input.cnf inputunsat.ass
Dimacs file: input.cnf
p cnf 10 30
3 20 -10 5
2 -2 -3
4 -4 -8 9 30
3 22 -11 15
2 -21 -9
4 -14 -18 19 -30
4 -24 -28 29 30
3 12 -21 25
2 -1 -29
4 -24 -28 29 -10
Assignment file: inputunsat.ass
1 -2 3 4 5 -6 -7 -8 9 -10 -11 12 -13 -14 15 -16 -17 -18 -19 -20
 21 -22 -23 -24 -25 -26 -27 -28 -29 30
Assignment is not satisfying the formula
```

**Note**:

- Scaricare il file `esercizio2.cpp`, modificarlo per inserire il codice necessario per rispondere a questo esercizio. **Caricare il file sorgente risultato delle vostre modifiche a soluzione di questo esercizio** nello spazio apposito.
- All'interno di questo programma **non è ammesso** l'utilizzo di variabili globali o di tipo `static` e di funzioni di libreria al di fuori di quelle definite in `iostream`, `fstream`, `cstdlib`, `sstream`.
- Si ricorda che, gli esempi di esecuzione sono puramente indicativi, e la soluzione proposta NON deve funzionare solo per l'input fornito, ma deve essere robusta a variazioni compatibili con la specifica riportata in questo testo.
- Si ricorda di inserire solo nuovo codice e di **NON MODIFICARE** il resto del programma (pena annullamento dell'esercizio).

esercizio2.cpp

input.cnf

inputsat.ass

inputunsat.ass

*Information for graders:*

(2) **Esercizio 2 v1**

ESSAY | marked out of 10 | penalty 0 | File picker

Il formato DIMACS CNF è una rappresentazione testuale di una formula in forma normale congiuntiva. Una formula in forma normale congiuntiva è una congiunzione (logica "and") di un insieme di clausole. Ogni clausola è una disgiunzione (logica "or") di un insieme di letterali. Un letterale è una variabile o una negazione di una variabile. DIMACS CNF utilizza gli **interi positivi** per rappresentare le variabili e la loro negazione (valore negativo) per rappresentare la corrispondente variabile negata. Un file DIMACS cnf contiene una intestazione (prima riga) così formata: `p cnf <numero clausole> <numero variabili>`, seguita da una serie di clausole. Ogni clausola inizia con un numero positivo che specifica quanti letterali ci sono nella clausola. Ad esempio, la formula $(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$ può essere rappresentata nel formato DIMACS CNF come segue:

```
p cnf 2 3
3 1 2 -3
3 -1 -2 3
```

Un assegnamento di valori alle variabili che rende la formula vera è detto **assegnamento soddisfacibile**. Un assegnamento consiste nell'assegnare un valore vero o falso ad ogni variabile. Ad esempio, l'assegnamento $\{x_1 = 1, x_2 = 0, x_3 = 1\}$ rende vera la formula sopra. Un assegnamento che rende la formula falsa è detto **assegnamento insoddisfacibile**. Una formula che non ha assegnamenti soddisfacibili è detta **insoddisfacibile**.

Un file DIMACS é rappresento in memoria con una struttura `Dimacs` che contiene il numero di clausole, il numero di variabili e le clausole rappresentati con una lista concatenata `cnf_t`, che a sua volta contiene una lista di letterali rapresentati con una lista concatenata `List` di interi (Si veda il file `esercizio2.cpp`).

Un assegnamento é rappresentato in memoria con una struttura `Assignment` che contiene un array di booleani tale che l'elemento `i` dell'array rappresenta il valore dell'assegnamento della variabile `i+1` (1 vero, 0 falso).

Il programma in `esercizio2.cpp` prende come argomento a) un file che rappresenta una formula in formato DIMACS e costruisce e popola la struttura `Dimacs`, b) un file che rappresenta un assegnamento parziale alle variabili della formula e costruisce e popola la struttura `Assignment`, e c) chiama una funzione `check_assigment` (da definire) che prende come primo argomento una struttura `Dimacs` e come secondo argomento una struttura `Assignment`, entrambe passate per *riferimento* e stampa a video `Assignment is satisfying the formula` se `check_assigment` ritorna `true`, altrimenti stampa `Assignment is not satisfying the formula`.

La funzione `check_assigment` ritorna `true` se e solo se l'assegnamento é un *assegnamento insoddisfacibile*.

La funzione `check_assigment` **deve essere ricorsiva** e **NON deve contenere iteratori** espliciti (`for`, `while`, `do-while`). La funzione `check_assigment` può ovviamente contenere codice sequenziale o condizionale. Sono consentite (se ritenute necessarie) chiamate a funzioni ricorsive ausiliarie che a loro volta **non contengano iterazioni esplicite** (`for`, `while`, `do-while`).

Il file `esercizio2.cpp` contiene tutto quanto necessario tranne la dichiarazione e la definizione della funzione `check_assigment`. Di seguito è riportato un esempio di esecuzione.

```
computer > ./a.out input.cnf inputunsat.ass
Dimacs file: input.cnf
```

```
p cnf 10 30
3 20 -10 5
2 -2 -3
4 -4 -8 9 30
3 22 -11 15
2 -21 -9
4 -14 -18 19 -30
4 -24 -28 29 30
3 12 -21 25
2 -1 -29
4 -24 -28 29 -10
Assignment file: inputunsat.ass
1 -2 3 4 5 -6 -7 -8 9 -10 -11 12 -13 -14 15 -16 -17 -18 -19 -20
 21 -22 -23 -24 -25 -26 -27 -28 -29 30
Assignment is satisfying the formula
computer > ./a.out input.cnf inputsat.ass
Dimacs file: input.cnf
p cnf 10 30
3 20 -10 5
2 -2 -3
4 -4 -8 9 30
3 22 -11 15
2 -21 -9
4 -14 -18 19 -30
4 -24 -28 29 30
3 12 -21 25
2 -1 -29
4 -24 -28 29 -10
Assignment file: inputsat.ass
1 -2 3 4 5 -6 -7 -8 -9 -10 -11 12 -13 -14 15 -16 -17 -18 -19 -20
 21 -22 -23 -24 -25 -26 -27 -28 -29 30
Assignment is not satisfying the formula
```

**Note**:

- Scaricare il file `esercizio2.cpp`, modificarlo per inserire il codice necessario per rispondere a questo esercizio. **Caricare il file sorgente risultato delle vostre modifiche a soluzione di questo esercizio** nello spazio apposito.
- All'interno di questo programma **non è ammesso** l'utilizzo di variabili globali o di tipo `static` e di funzioni di libreria al di fuori di quelle definite in `iostream`, `fstream`, `cstdlib`, `sstream`.
- Si ricorda che, gli esempi di esecuzione sono puramente indicativi, e la soluzione proposta NON deve funzionare solo per l'input fornito, ma deve essere robusta a variazioni compatibili con la specifica riportata in questo testo.
- Si ricorda di inserire solo nuovo codice e di **NON MODIFICARE** il resto del programma (pena annullamento dell'esercizio).

esercizio2.cpp

input.cnf

inputsat.ass

inputunsat.ass

*Information for graders:*

*Total of marks: 20*