The University of Melbourne
Department of Computing and Information Systems
Semester 2 Assessment 2017

COMP90046 Constraint Programming

Mid Semester Test: Sample Answers

Time Allowed: 50 minutes

Student Number:	

Question	Mark
1	
2	
3	
4	

Authorized materials: Books and calculators are not permitted.

Instructions to students:

This mid semester test counts for 0% of your final grade. It is used for self assessment, and for supporting a case for passing the subject if your assignment and final exam marks are borderline. There are 8 pages and 4 questions for a total of 12 marks. Attempt to answer all of the questions. Values are indicated for each question and subquestion — be careful to allocate your time according to the value of each question.

Question 1 [2 marks]

Write a possible output from the following MiniZinc models

(a)
```
var 0..3: x;
var 1..4: y;
constraint x = y;
constraint x = 2 \/ y = 3;
solve satisfy;
```
`x = 2; y = 2;` or `x = 3; y = 3;`

(b)
```
var 0..3: x;
var 1..4: y;
constraint    x >= 1
           /\ x <= 2 -> y = 3;
solve minimize x+y;
```
`x = 0; y = 1;`

(c)
```
array[1..3] of var 0..2: x;
constraint x[2] != x[1] /\ x[2] != x[3];
constraint x[1] < x[3];
solve satisfy;
```
`x = [0,2,1]` or `x = [1,0,2]` or `x = [0,1,2]`

(d)
```
array[1..4] of var 0..4: x;
constraint alldifferent(x);
constraint x[1] = x[3];
solve minimize sum(x);
```
`====UNSATISFIABLE====`

Question 2 [3 marks]

Aaron, Betty, Charlie, Debbie, and Eric each ordered a pizza with three of the following five toppings: green pepper, mushrooms, onions, pepperoni, and sausage. The only topping that Aaron and Charlie had in common was sausage. The only topping Debbie and Eric had in common was pepperoni. The only topping Charlie and Betty had in common was mushrooms. The only topping Debbie and Betty had in common was green pepper.

Write a MiniZinc model to determine which pizzas each person ordered.

```
enum TOPPING = { Pepper, Mushroom, Onion, Pepperoni, Sausage };
var set of TOPPING: Aaron;
var set of TOPPING: Betty;
var set of TOPPING: Charlie;
var set of TOPPING: Debbie;
var set of TOPPING: Eric;
constraint Aaron intersect Charlie = { Sausage };
constraint Debbie intersect Eric = { Pepperoni };
constraint Charlie intersect Betty = { Mushroom };
constraint Debbie intersect Betty = { Pepper };
constraint card(Aaron) = 3;
constraint card(Betty) = 3;
constraint card(Charlie) = 3;
constraint card(Debbie) = 3;
constraint card(Eric) = 3;
solve satisfy;
```

[3 marks]

Question 3 [3 marks]

Give the best MiniZinc representation for each of the following decisions, including all constraints

(a) Choosing a set of at most 3 elements of numbers from 1..100000.

```
array[1..3] of var 0..10000: x;
constraint x[1] >= x[2] + (x[1] != 0);
constraint x[2] >= x[3] + (x[2] != 0);
```

(b) Choosing a set of at least 5 elements from the numbers 1..12.

```
var set of 1..12: x;
constraint card(x) >= 5;
```

(c) Choosing a subset of 1..20 which adds up to 30; Any representation is probably OK, maybe the best is

```
array[1..20] of var 0..1: x;
constraint sum(i in 1..20)(x[i]*i) = 30;
```

alternatively

```
var set of 1..20: x;
constraint sum(i in x)(*i) = 30;
```

Question 4 [4 marks]

Given a set of prize winners to order for a prize giving ceremony, there are certain protocols that need to be followed. No two recipients of the same prize should be next to each other in the order.

A best order is closest to the order of increasing age, that is we want to minimize the number of times that for any two winners in the order they appear in the ceremony, the first is older than the second.

Consider winners, A, B, C, D, E, F, G, H with prizes [Maths, English, Maths, Latin, Latin, English, Drama, Drama] and ages [12, 15, 12, 9, 11, 10, 17, 18]. A MiniZinc data file encoding this information for the model below is:

```
WINNER = {A,B,C,D,E,F,G,H};
PRIZE = {Maths, English, Latin, Drama};
recieves = [Maths, English, Maths, Latin, Latin, English, Drama, Drama];
age = [12, 15, 12, 9, 11, 10, 17, 18];
```

A solution is the order [D,F,E,A,B,G,C,H], which has order of prize [Latin, English, Latin, Maths, English, Drama, Maths, Drama], note no two same prizes are adjacent, and age order is [9, 10, 11, 12, 15, 17, 12, 19]. The objective is 2 since C appears after B and G which are both older.

Given MiniZinc starting model of the form

```
enum WINNER;
enum PRIZE;
array[WINNER] of PRIZE: recieves;
array[WINNER] of int: age;

int: n = card(WINNER);
set of int: POS = 1..n;
array[POS] of var WINNER: person;
```

(a) Complete the model. You do not have to rewrite the part above. [2 marks]

```
enum WINNER;
enum PRIZE;
array[WINNER] of PRIZE: receives;
array[WINNER] of int: age;
int: n = card(WINNER);
set of int: POS = 1..n;
array[POS] of var WINNER: person;
include "alldifferent.mzn";
constraint alldifferent(person);
```

```
constraint forall(p in 1..n-2)
                 (receives[person[p]] != receives[person[p+1]]);
var int: obj;
constraint obj = sum(p1,p2 in POS where p1 < p2)
                    (age[person[p1]] > age[person[p2]]);
solve maximize obj;
```

(b) Actually there is a missing constraint. The last person recieving prize p1 should not appear before the first person recieving another prize p2, for all pairs of prizes p1, p2; otherwise it seems the first kind of prize p1 is more important that the second p2.

The previous solution is invalid since e.g. the last person receiving a Latin or English prize appears before the first Drama prize.

Modify your model to include this constraint. [2 marks]

```
array[WINNER] of var POS: posn;
include "inverse.mzn";
constraint inverse(posn, person);
constraint forall(z1, z2 in PRIZE where z1 != z2)
                 (max([ posn[w] | w in WINNER where receives[w] = z1]) >=
                  min([ posn[w] | w in WINNER where receives[w] = z2]));
```