Project 2 (Futoshiki)
CS 4613 Artificial Intelligence
Professor Edward Wong
Gina Joerger
May 6, 2020

How To Run The Program:

To run the program, simply run the code provided in the Futoshiki.txt file in a python compiler. To change which input text document will be addressed in the program, the code in the main function will have to be rewritten, so instead of document = 'Input1.txt', another document can be called. To have an output with a different name than Output1.txt, the line of code stating sys.stdout = open('Output1.txt','wt') will also have to be adjusted. These two lines are both in the main() function.

Source Code:

```
import functools
import re
import sys
import queue

#Specifies a constraint class
class constraint:
    def __init__(self, d, name):
        self.domain = [i for i in d]
        self.name = name

#Specifies a Unary Constraint Class
class UnaryConstraint:
    def __init__(self, v, func):
        self.var = v
        self.func = func

#Specifies a Binary Constraint Class
class BinaryConstraint:
    def __init__(self, v1, v2, func):
        self.var1 = v1
        self.var2 = v2
        self.func = func

#Side function for lamda for "="
def side(x, item):
    return x == item

#Making sure all items based on the constraints are different
def notSame(constraints, oneLine):
```

```
    func = lambda x, y: x != y
    for a in range(len(oneLine)):
        for b in range(len(oneLine)):
            if (a != b):
                constraints.append(BinaryConstraint(oneLine[a], oneLine[b], func))

#Forward Checking
def forwardChecking(document):
        unary = []
        binary = []
        change = []

        #Get the input into a processable format
        lines = open(document).readlines()
        for i in range(16):
            l = lines[i]

            #Definitely cleaner way to do this, I just wanted to spend more time on the algorithm
            if i == 0:
                for x in range(len(l)):
                    if x == 0 or x == 2 or x == 4 or x == 6 or x == 8:
                        if l[x] != '0' and x == 0:
                            change.append("A1="+l[x])
                        elif l[x] != '0' and x == 2:
                            change.append("A2="+l[x])
                        elif l[x] != '0' and x == 4:
                            change.append("A3="+l[x])
                        elif l[x] != '0' and x == 6:
                            change.append("A4="+l[x])
                        elif l[x] != '0' and x == 8:
                            change.append("A5="+l[x])
            elif i == 1:
                for x in range(len(l)):
                    if x == 0 or x == 2 or x == 4 or x == 6 or x == 8:
                        if l[x] != '0' and x == 0:
                            change.append("B1="+l[x])
                        elif l[x] != '0' and x == 2:
                            change.append("B2="+l[x])
                        elif l[x] != '0' and x == 4:
                            change.append("B3="+l[x])
                        elif l[x] != '0' and x == 6:
                            change.append("B4="+l[x])
                        elif l[x] != '0' and x == 8:
                            change.append("B5="+l[x])
            elif i == 2:
                for x in range(len(l)):
```

```
        if x == 0 or x == 2 or x == 4 or x == 6 or x == 8:
            if l[x] != '0' and x == 0:
                change.append("C1="+l[x])
            elif l[x] != '0' and x == 2:
                change.append("C2="+l[x])
            elif l[x] != '0' and x == 4:
                change.append("C3="+l[x])
            elif l[x] != '0' and x == 6:
                change.append("C4="+l[x])
            elif l[x] != '0' and x == 8:
                change.append("C5="+l[x])
    elif i == 3:
        for x in range(len(l)):
            if x == 0 or x == 2 or x == 4 or x == 6 or x == 8:
                if l[x] != '0' and x == 0:
                    change.append("D1="+l[x])
                elif l[x] != '0' and x == 2:
                    change.append("D2="+l[x])
                elif l[x] != '0' and x == 4:
                    change.append("D3="+l[x])
                elif l[x] != '0' and x == 6:
                    change.append("D4="+l[x])
                elif l[x] != '0' and x == 8:
                    change.append("D5="+l[x])
    elif i == 4:
        for x in range(len(l)):
            if x == 0 or x == 2 or x == 4 or x == 6 or x == 8:
                if l[x] != '0' and x == 0:
                    change.append("E1="+l[x])
                elif l[x] != '0' and x == 2:
                    change.append("E2="+l[x])
                elif l[x] != '0' and x == 4:
                    change.append("E3="+l[x])
                elif l[x] != '0' and x == 6:
                    change.append("E4="+l[x])
                elif l[x] != '0' and x == 8:
                    change.append("E5="+l[x])
    elif i == 6:
        for x in range(len(l)):
            if x == 0 or x == 2 or x == 4 or x == 6 or x == 8:
                if l[x] != '0' and x == 0:
                    change.append("A1"+l[x]+"A2")
                elif l[x] != '0' and x == 2:
                    change.append("A2"+l[x]+"A3")
                elif l[x] != '0' and x == 4:
                    change.append("A3"+l[x]+"A4")
```

```
            elif l[x] != '0' and x == 6:
                change.append("A4"+l[x]+"A5")
    elif i == 7:
        for x in range(len(l)):
            if x == 0 or x == 2 or x == 4 or x == 6 or x == 8:
                if l[x] != '0' and x == 0:
                    change.append("B1"+l[x]+"B2")
                elif l[x] != '0' and x == 2:
                    change.append("B2"+l[x]+"B3")
                elif l[x] != '0' and x == 4:
                    change.append("B3"+l[x]+"B4")
                elif l[x] != '0' and x == 6:
                    change.append("B4"+l[x]+"B5")
    elif i == 8:
        for x in range(len(l)):
            if x == 0 or x == 2 or x == 4 or x == 6 or x == 8:
                if l[x] != '0' and x == 0:
                    change.append("C1"+l[x]+"C2")
                elif l[x] != '0' and x == 2:
                    change.append("C2"+l[x]+"C3")
                elif l[x] != '0' and x == 4:
                    change.append("C3"+l[x]+"C4")
                elif l[x] != '0' and x == 6:
                    change.append("C4"+l[x]+"C5")
    elif i == 9:
        for x in range(len(l)):
            if x == 0 or x == 2 or x == 4 or x == 6 or x == 8:
                if l[x] != '0' and x == 0:
                    change.append("D1"+l[x]+"D2")
                elif l[x] != '0' and x == 2:
                    change.append("D2"+l[x]+"D3")
                elif l[x] != '0' and x == 4:
                    change.append("D3"+l[x]+"D4")
                elif l[x] != '0' and x == 6:
                    change.append("D4"+l[x]+"D5")
    elif i == 10:
        for x in range(len(l)):
            if x == 0 or x == 2 or x == 4 or x == 6 or x == 8:
                if l[x] != '0' and x == 0:
                    change.append("E1"+l[x]+"E2")
                elif l[x] != '0' and x == 2:
                    change.append("E2"+l[x]+"E3")
                elif l[x] != '0' and x == 4:
                    change.append("E3"+l[x]+"E4")
                elif l[x] != '0' and x == 6:
                    change.append("E4"+l[x]+"E5")
```

```
elif i == 12:
    for x in range(len(l)):
        if x == 0 or x == 2 or x == 4 or x == 6 or x == 8:
            if l[x] == '^' and x == 0 :
                change.append("A1"+"<"+"B1")
            elif l[x] == '^' and x == 2:
                change.append("A2"+"<"+"B2")
            elif l[x] == '^' and x == 4:
                change.append("A3"+"<"+"B3")
            elif l[x] == '^' and x == 6:
                change.append("A4"+"<"+"B4")
            elif l[x] == '^' and x == 8:
                change.append("A5"+"<"+"B5")
            elif l[x] == 'v' and x == 0 :
                change.append("A1"+">"+"B1")
            elif l[x] == 'v' and x == 2:
                change.append("A2"+">"+"B2")
            elif l[x] == 'v' and x == 4:
                change.append("A3"+">"+"B3")
            elif l[x] == 'v' and x == 6:
                change.append("A4"+">"+"B4")
            elif l[x] == 'v' and x == 8:
                change.append("A5"+">"+"B5")
elif i == 13:
    for x in range(len(l)):
        if x == 0 or x == 2 or x == 4 or x == 6 or x == 8:
            if l[x] == '^' and x == 0 :
                change.append("B1"+"<"+"C1")
            elif l[x] == '^' and x == 2:
                change.append("B2"+"<"+"C2")
            elif l[x] == '^' and x == 4:
                change.append("B3"+"<"+"C3")
            elif l[x] == '^' and x == 6:
                change.append("B4"+"<"+"C4")
            elif l[x] == '^' and x == 8:
                change.append("B5"+"<"+"C5")
            elif l[x] == 'v' and x == 0 :
                change.append("B1"+">"+"C1")
            elif l[x] == 'v' and x == 2:
                change.append("B2"+">"+"C2")
            elif l[x] == 'v' and x == 4:
                change.append("B3"+">"+"C3")
            elif l[x] == 'v' and x == 6:
                change.append("B4"+">"+"C4")
            elif l[x] == 'v' and x == 8:
                change.append("B5"+">"+"C5")
```

```
elif i == 14:
    for x in range(len(l)):
        if x == 0 or x == 2 or x == 4 or x == 6 or x == 8:
            if l[x] == '^' and x == 0 :
                change.append("C1"+"<"+"D1")
            elif l[x] == '^' and x == 2:
                change.append("C2"+"<"+"D2")
            elif l[x] == '^' and x == 4:
                change.append("C3"+"<"+"D3")
            elif l[x] == '^' and x == 6:
                change.append("C4"+"<"+"D4")
            elif l[x] == '^' and x == 8:
                change.append("C5"+"<"+"D5")
            elif l[x] == 'v' and x == 0 :
                change.append("C1"+">"+"D1")
            elif l[x] == 'v' and x == 2:
                change.append("C2"+">"+"D2")
            elif l[x] == 'v' and x == 4:
                change.append("C3"+">"+"D3")
            elif l[x] == 'v' and x == 6:
                change.append("C4"+">"+"D4")
            elif l[x] == 'v' and x == 8:
                change.append("C5"+">"+"D5")
elif i == 15:
    for x in range(len(l)):
        if x == 0 or x == 2 or x == 4 or x == 6 or x == 8:
            if l[x] == '^' and x == 0 :
                change.append("D1"+"<"+"E1")
            elif l[x] == '^' and x == 2:
                change.append("D2"+"<"+"E2")
            elif l[x] == '^' and x == 4:
                change.append("D3"+"<"+"E3")
            elif l[x] == '^' and x == 6:
                change.append("D4"+"<"+"E4")
            elif l[x] == '^' and x == 8:
                change.append("D5"+"<"+"E5")
            elif l[x] == 'v' and x == 0 :
                change.append("D1"+">"+"E1")
            elif l[x] == 'v' and x == 2:
                change.append("D2"+">"+"E2")
            elif l[x] == 'v' and x == 4:
                change.append("D3"+">"+"E3")
            elif l[x] == 'v' and x == 6:
                change.append("D4"+">"+"E4")
            elif l[x] == 'v' and x == 8:
                change.append("D5"+">"+"E5")
```

```
l = ','.join(change)

#Identify initial rows and columns
r = []
c = []
for i in range(5):
        r.append(chr(ord('A') + i))
        c.append(chr(ord('1') + i))
names = [a + b for a in r for b in c]

#Get all domains
domain = []
for i in range(5):
        domain.append(i+1)

#Each item made into a constraint variable
items = dict()
for uni in names:
        items[uni] = constraint(domain, uni)

#Get each requirement in the game
req = re.findall('\w+\W+\w+',l)

#For each, separate apart the variables, operator, and values
for i in req:
        #For < or >
        if re.findall('\w+\d+<\w+\d+', i) or re.findall('\w+\d+>\w+\d+', i):
                left = re.findall('^\w+\d+', i)[0]
                right = re.findall('\w+\d+$', i)[0]
                operator = re.findall('\W', i)[0]

                #Convert inequalities to lambda, make them binary constraints
                if operator == '>':
                   func = lambda x,y: x > y
                   func2 = lambda x,y: x < y
                elif operator == '<':
                   func = lambda x,y: x < y
                   func2 = lambda x,y: x > y
                binary.append(BinaryConstraint(items[left], items[right], func))
                binary.append(BinaryConstraint(items[right], items[left], func2))
        else:
        #For = operator
                if re.findall('\w+\d+=\d+', i):
                        var = re.findall('^\w+\d+', i)[0]
                        item = re.findall('\d+$', i)[0]
                elif re.findall('\d+=\w+\d+', i):
```

```
                    var = re.findall('\w+\d+$', i)[0]
                    item = re.findall('^\d+$', i)[0]

                #Convert equalities to lamda functions
                func = functools.partial(side, item = int(item))
                unary.append(UnaryConstraint(items[var], func))

    #Establish difference constraint for each row
    for i in r:
        oneLine = []
        for k in items.keys():
            if (str(k).startswith(i)):
                oneLine.append(items[k]) #All constraints contained in row i
        notSame(binary, oneLine) #notSame constraints for row elements

    #Establish difference constraint for each column
    for i in c:
        oneLine = []
        for k in items.keys():
            key = str(k)
            if (key[1] == i):
                oneLine.append(items[k]) #All constraints contained in column i
        notSame(binary, oneLine)
    return items, unary, binary

#Backtracking
def bt(items, constraints):
    for i in constraints[0]:
        domain = list(i.var.domain)
        for x in domain:
            if not i.func(x):
                i.var.domain.remove(x)
    return bt2({}, items, constraints)

#Backtracking Pt. 2 with inferences
def bt2(assignment, items, constraints):
    if len(assignment) == 25:
        return assignment
    variable = mrv(assignment, items)
    for item in variable.domain:
        assignment[variable.name] = item

        #Inference beginning
        infer = {}
        x = []
        for i in variable.domain:
```

```
    if i != item:
        x.append(i)
variable.domain = [item]
infer[variable.name] = x
q = queue.Queue()

#Binary Constraint Variable into Queue
for c in constraints[1]:
    if c.var2.name == variable.name:
        q.put(c)
while not q.empty():
    limit = q.get()

    #If inconsistent, remove
    remove = []
    if isinstance(limit, UnaryConstraint):
        dom = list(limit.var.domain)
        for x in dom:
            if not limit.func(x):
                limit.var.domain.remove(x)
        return True
    elif isinstance(limit, BinaryConstraint):
        domain1 = list(limit.var1.domain)
        domain2 = list(limit.var2.domain)

        #Var1 Domain
        for x in domain1:
            go = True
            #Var2 Domain
            for y in domain2:
                if x == y:
                    continue
                #If Var2 Domain is wrong, remove
                if True == limit.func(x, y):
                    go = False
                    break
            if go:
                remove.append(x)
                limit.var1.domain.remove(x)
    if remove:
        for i in constraints[1]:
            if i.var2.name == limit.var1.name:
                q.put(i)
        if limit.var1.name not in infer.keys():
            infer[limit.var1.name] = remove
        else:
```

```
                infer[limit.var1.name] += remove
            if not limit.var1.domain:
                infer['x'] = True
                break

        #Continue Backtracking
        if 'x' not in infer.keys():
            result = bt2(assignment, items, constraints)
            if result:
                return result
        assignment.pop(variable.name)
        if 'x' in infer.keys():
            infer.pop('x')
        for i in infer.keys():
            items[i].domain += infer[i]
    return False

#Minimum Remaining Values for Backtracking
def mrv(assignment, items):
    num = 25
    l = None
    for i in items.keys():
        if i not in assignment and len(items[i].domain) < num:
            l = items[i]
            num = len(items[i].domain)
    return l

def main():
    sys.stdout = open('Output1.txt','wt') #Makes output txt file for writing
    document = 'Input1.txt'
    items, unary, binary = forwardChecking(document) #Forward Checking
    constraints = [unary, binary] #sets constraints
    final = bt(items, constraints) #through backtracking in MRV, gets final results

    #Prints out result in required format
    x = 0
    for k in sorted(final.keys()):
        print(final[k], '', end="")
        x += 1
        if x % 5 == 0:
            print('')

main()
```

Output 1

2 1 5 4 3
1 3 4 2 5
4 5 1 3 2
5 2 3 1 4
3 4 2 5 1

Output 2

3 4 2 5 1
1 5 4 2 3
2 3 5 1 4
5 1 3 4 2
4 2 1 3 5

Output 3

3 1 5 2 4
5 2 3 4 1
1 3 4 5 2
4 5 2 1 3
2 4 1 3 5