CS106X Instructor: Cynthia Lee

Autumn 2015 Practice Exam

PRACTICE MIDTERM EXAM – SOLUTIONS

---

1. **ADTs (30pts).**

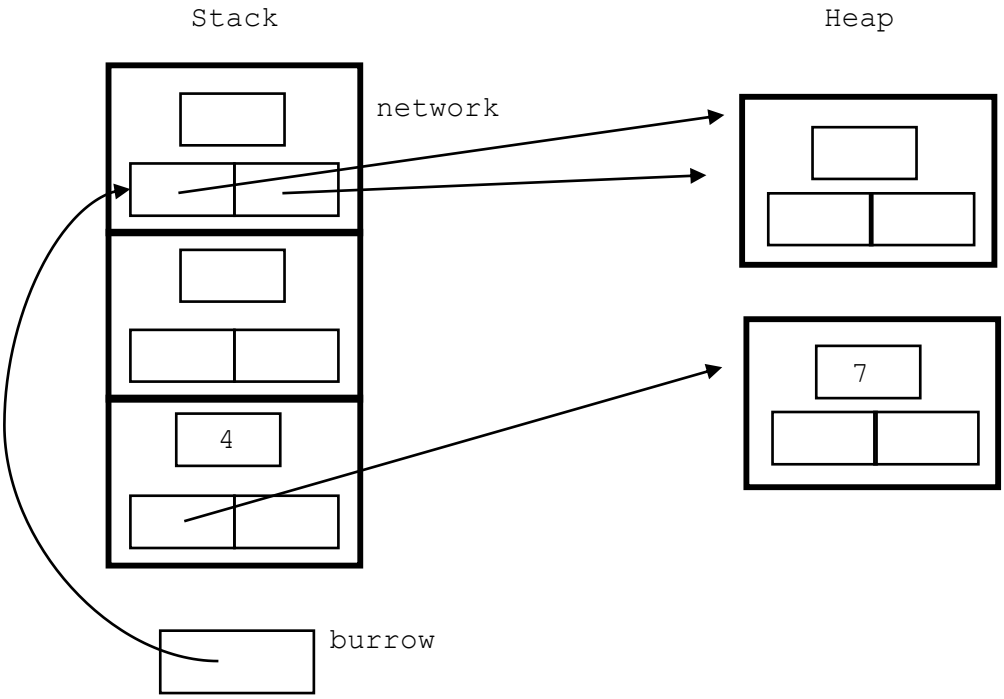
```
bool canFit(Grid<bool>& puzzle, Grid<bool> piece) {
  for (int turn=0; turn<4; turn++){
     for (int row=0; row<board.numRows()-piece.numRows()+1; row++){
        for (int col=0; col<puzzle.numCols()-piece.numCols()+1; col++){
           bool foundConflict = false;
           for (int i=0; i<piece.numRows(); i++){
              for (int j=0; j<piece.numCols(); j++){
                 if (piece[i][j] && puzzle[row+i][col+j]){
                    foundConflict = true;
                    break;
                 }
              }
           }
           if (!foundConflict){
              for (int i=0; i<piece.numRows(); i++){
                 for (int j=0; j<piece.numCols(); j++){
                    if (piece[i][j]) puzzle[row+i][col+j] = true;
                 }
              }
              return true;
           }
        }
     }
     rotate90(piece);
  }
  return false;
}

void rotate90(Grid<bool>& piece) {
  Grid<bool> newpiece(piece.numCols(),piece.numRows());
     for (int r=0; r<piece.numRows(); r++)
        for (int c=0; c<piece.numCols(); c++)
           newpiece[c][piece.numRows()-1-r] = piece[r][c];
  piece = newpiece;
}
```

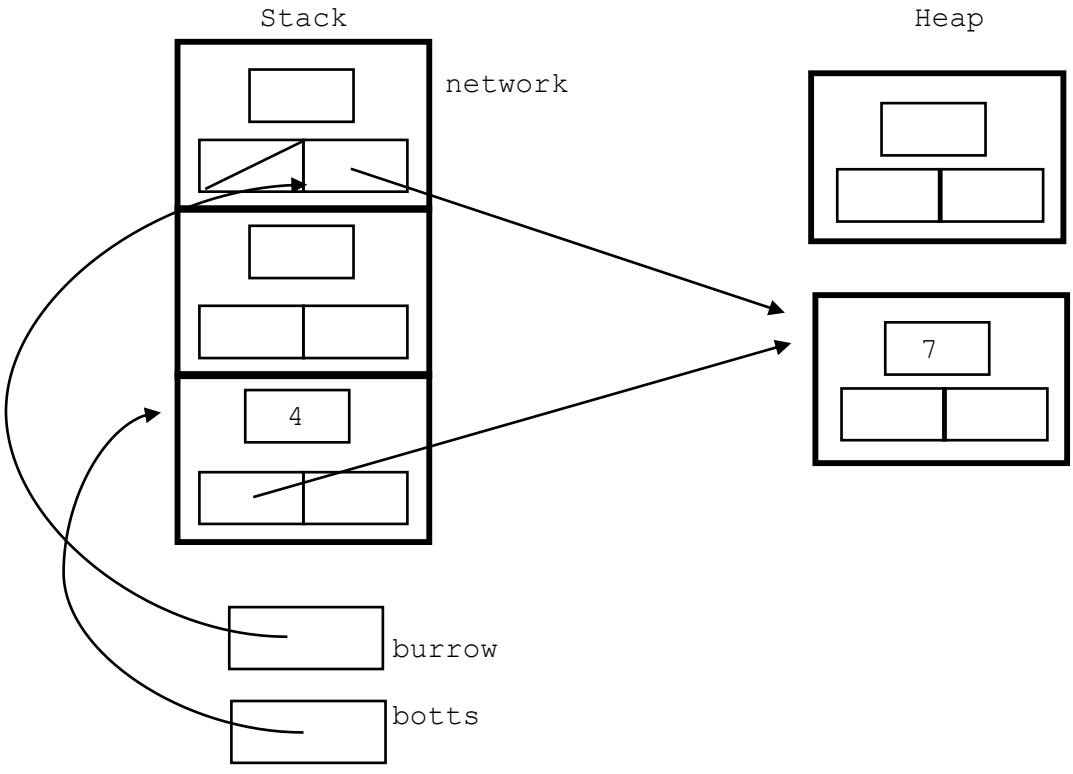
---

2. **Pointers and Memory (21pts).**

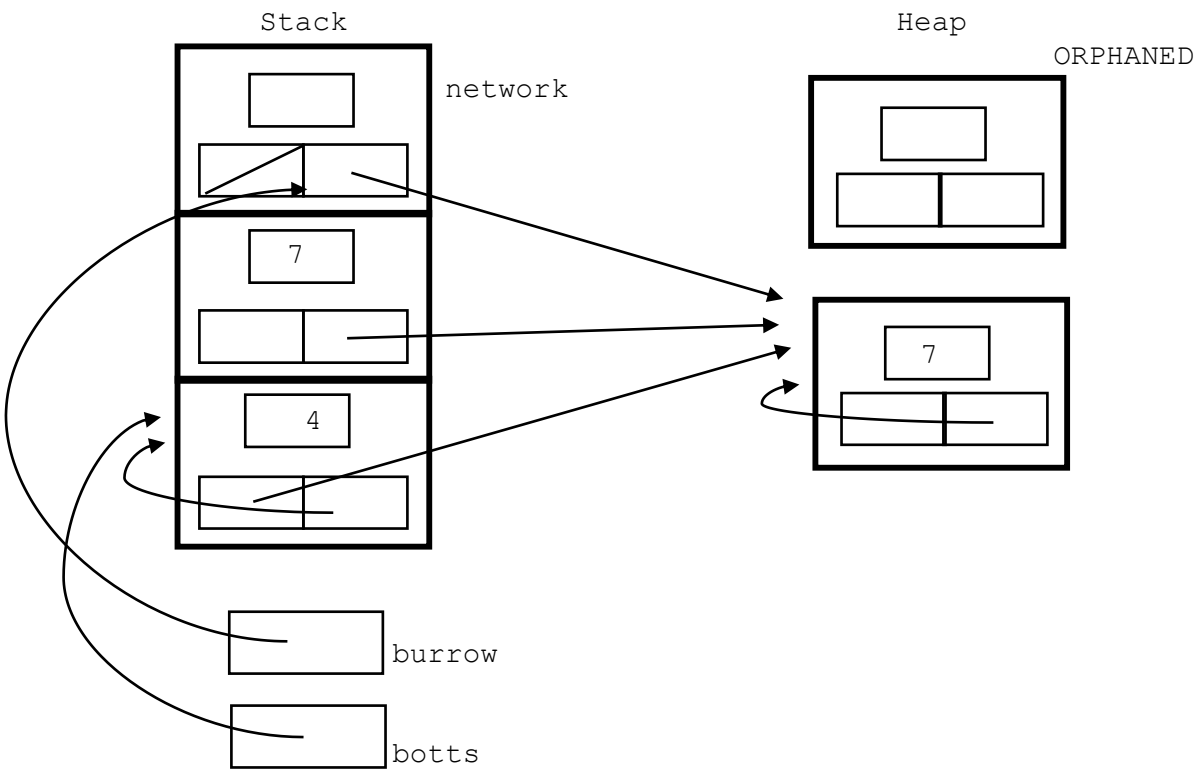
DRAWING #1:



DRAWING #2:



DRAWING #3:



---

3. **Classes (30pts).**

```
// in farm.h file

   private:
      struct Animal {
             string type;
             int cost;
      };
      int capacity;
      Vector<Animal> animals;
   };
//end of farm.h file


//beginning of farm.cpp file
#include “farm.h”

Farm::Farm(int capacity) {
       this->capacity = capacity;
}

Farm::~Farm() { }
```

```
Vector<string> Farm::getAnimalTypes() {
	Vector<string> noDupes;
	Set<string> types;
	for(Animal a : animals) types.add(a.type);
	for(string type : types) noDupes.add(type);
	return noDupes;
}

int Farm::getTotalNumberOfAnimals() {
	return animals.size();
}

int Farm::getTotalCost() {
	int totalCost = 0;
	for (Animal a : animals) totalCost += a.cost;
	return totalCost;
}


void Farm::addAnimal(string type, int cost) {
	if (capacity == animals.size()) error("No more room on farm");
	Animal newAnimal;
	newAnimal.type = type;
	newAnimal.cost = cost;
	animals.add(newAnimal);
}
void Farm::removeAnimal(string type) {
	for (int i = 0; i < animals.size(); i++) {
		if (animals[i].type == type) {
			animals.remove(i);
			return;
		}
	}
	error ("Animal of that type does not exist on this farm");
}

int Farm::getNumberOfAnimalType(string type) {
	int count = 0;
	for (Animal a : animals) {
		if (a.type == type) count++;
	}
	return count;
}
```

---

4. **Recursion (30pts). Recursive Backtracking (30pts).**

```
static bool canWrite(string str, unsigned int n, map<int, int> &useCount) {
	if(str.length() == 0) return true;
	if(isspace(str[0])) return canWrite(str.substr(1), n, useCount);
	for(int i = 0; i < 16; i++) {
		if(useCount[i] < n && kStandardCubes[i].find(str[0]) != string::npos) {
			useCount[i]++;
			if(canWrite(str.substr(1), n, useCount)) return true;
			useCount[i]--;
		}
	}
	return false;
}

static bool canWriteInCubes(string str, unsigned int n) {
	map<int, int> useCount;
	return canWrite(str, n, useCount);
}
```

---

5. **Big-O (9pts).**

$O(N^2)$
$O(N^2)$
$O(N)$