CMPT 225 - Midterm 1
October 13, 2017
Time: 40 minutes

DO NOT begin until told to do so.
Fill in your name and student ID below.
Turn off your phone and put it in your bag.
Keep your student ID card on your desk at all times.
You can't have anything other than a pen/pencil and your student ID card on you during the test.
Before you begin please check that your exam consists of **6** consecutively numbered questions.
Read the questions carefully and more than once.
Good luck!

NAME: …….....

STUDENT ID: ……

SCORE:

Question	**Marks**
Question 1 (6 marks)	
Question 2 (10 marks)	
Question 3 (6 marks)	
Question 4 (6 marks)	
Question 5 (12 marks)	
Question 6 (10 marks)	

1. my_queue implements ADT Queue. What does the following piece of code write to the standard output? (6 marks)

```
my_queue<int> q;
q.enqueue(5);
q.enqueue(6);
q.enqueue(7);
cout << "q.front() is " << q.front() << endl;
q.dequeue();
q.dequeue();
cout << "q.front() is " << q.front() << endl;
cout << "q.size() is " << q.size() << endl;
```

```
q.front() is 5
q.front() is 7
q.size() is 1
```

2. Write a **formal interface** (an abstract class declaring all the functionality) for a **generic** bag ADT with the following functions. Do not worry about exceptions. (10 marks)

ADT functions:
Put something in – add(item)
Take an item out – remove(item)
Count how many things are in it – getFrequencyOf(item)
See if it is empty – isEmpty()
Count the items in it – getCurrentSize()

```
template <typename E>
class Bag{
  public:
      virtual void add(E item)=0;
      void remove(E item);
      int getFrequencyOf(E item) const;
      int isEmpty() const;
      int getCurrentSize() const;
};
```

3. Give the code to delete an element from a Doubly Linked List. Assume the pointer is pointing to a valid node and don't worry about exceptions. The signature is void remove(DNode* p);

(6 marks)

```
void remove(DNode* p){
        p->next->prev = p->prev;
        p->prev->next = p->next;
        p->prev = NULL;
        p->next = NULL;
        delete p;
        n--;
}
```

4. What is the time complexity of the following pieces of code in big-O notation? You don't need to prove or justify your answer.

a. (4 marks)

```
// Returns the sum of n integer in A starting at index i
int BinarySum( int * A, int i, int n){
        if (n == 1)
                return A[i];
        return BinarySum(A, i, ⌈n/2⌉) + BinarySum(A, i+⌈n/2⌉, ⌊n/2⌋);
}
```

b. (2 marks)

```
void enumerate(int n){
        for (int i=0; i< n; i++)
                for (int j= i-3; j< i+3; j++)
                        for (int k=j; k<n; k++)
                                cout << i << " " << j << " " << k << endl;
 }
```

a. O(n)
b.O(n^2)

5. Think about what we should do to evaluate an expression in postfix notation using stacks. As an example, I have done the stack operations needed to evaluate the expression expA. Do the same thing for the expressions expB. (12 marks)

stack operations needed to evaluate expA. 52 3 4 + + 3 2 * 4 / -	stack operations needed to evaluate expB. 24 3 / 2 + 3 * 5 -

stack<float> s;
s.push(52);
s.push(3);
s.push(4);
float a = s.top();
s.pop();
float b = s.top();
s.pop();
s.push(a + b);
a = s.top();
s.pop();
b = s.top();
s.pop();
s.push(a + b);
s.push(3);
s.push(2);
a = s.top();
s.pop();
b = s.top();
s.pop();
s.push(a * b);
s.push(4);
a = s.top();
s.pop();
b = s.top();
s.pop();
s.push(b/a);
a = s.top();
s.pop();
b = s.top();
s.pop();
s.push(b-a);
float result = s.top();
	stack<float> s;
	s.push(24);
	s.push(3);
	float a = s.top();
	s.pop();
	float b = s.top();
	s.pop();
	s.push(b/a);
	s.push(2);
	a = s.top();
	s.pop();
	b = s.top();
	s.pop();
	s.push(b+a);
	s.push(3);
	a = s.top();
	s.pop();
	b = s.top();
	s.pop();
	s.push(b*a);
	s.push(5);
	a = s.top();
	s.pop();
	b = s.top();
	s.pop();
	s.push(b-a);
	float result = s.top();

6. Write a short **recursive** C++ function that determines if a string s is a palindrome, that is, it is equal to its reverse. For example, "racecar" and "gohangasalamiimalasagnahog" are palindromes but "bird" is not. (10 marks)

```
bool isPalindrome(const string& s, int i, int j){
    if (i>=j)
      return true;
    if (s[i]!=s[j])
      return false;
    return isPalindrome(s, i+1, j-1);
}

bool isPalindrome(const string& s){
    return isPalindrome(s, 0, s.size()-1);
}
```