CSCI 3333 Homework: Lists
(with Solutions)

1 List Syntax

Problem 1. Write a declaration of a vector of floats named V.
Solution 1. `vector<float> V;`

Problem 2. Write a declaration of a vector of float pointers named P.
Solution 2. `vector<float*> P;`

Problem 3. Write a declaration of a vector of a vector of floats named VV.
Solution 3. `vector<vector<float>> VV;`

Problem 4. Write a C++ code snippet that creates a vector named ABC containing the elements "easy", "as", "1,2,3" (in that order).
Solution 4.
```cpp
vector<string> ABC;
ABC.push_back("easy");
ABC.push_back("as");
ABC.push_back("1,2,3");
```

2 Using Lists

Problem 5. Write a C++ function named `print_all` that takes a `vector<string>` named A and returns nothing. The function should print the elements of the `vector`, one per line.
Solution 5.
```cpp
void print_all(vector<string> V)
{
    for (int i = 0; i < V.size(); ++i)
        cout << V[i] << endl;
}
```

Problem 6. (Based on Book Exercise 3.5) Write a C++ function named `union` that takes two `vector<float>`s named A and B and returns a `vector<float>`. The returned vector should contain the union of the elements in A and B (i.e., contains exactly the elements in A and B with no duplicates).
Solution 6.
vector<float> union(vector<float> A, vector<float> B) {
    vector<float> U;
    for (int i = 0; i < A.size(); ++i) {
        bool found = false;
        for (int j = 0; j < U.size(); ++j) {
            if (U[j] == A[i])
                found = true;
        }
        if (!found)
            U.push_back(A[i]);
    }
    for (int i = 0; i < B.size(); ++i) {
        bool found = false;
        for (int j = 0; j < U.size(); ++j) {
            if (U[j] == B[i])
                found = true;
        }
        if (!found)
            U.push_back(B[i]);
    }
    return U;
}

Problem 7. (Based on Book Exercise 3.1) Write a C++ function named print_indices that takes two vector<int>s named A and I and returns nothing. The function should print the elements of A, one per line, at the valid indices in I (elements in I less than 0 and greater than A.size()-1 can be ignored). The elements may be printed in any order.

Solution 7. void print_indices(vector<int> A, vector<int> I) {
    for (int i = 0; i < I.size(); ++i)
        if (0 <= i && i < A.size())
            cout << A[i] << endl;
}
3 List Asymptotics

Problem 8. Give the worst-case running times of the following operations of a dynamic-array-based (e.g., C++ STL’s vector) list containing $n$ elements.

- Adding an element:
  - To the back: $\Theta(n)$.
  - To the front: $\Theta(n)$.
  - To the middle: $\Theta(n)$.

- Removing an element:
  - From the back: $\Theta(1)$.
  - From the front: $\Theta(n)$.
  - From the middle: $\Theta(n)$.

- Accessing an element:
  - At the back: $\Theta(1)$.
  - At the front: $\Theta(1)$.
  - At the middle: $\Theta(1)$.

Solution 8. Worst-case running times:

- Adding an element:
  - To the back: $\Theta(n)$.
  - To the front: $\Theta(n)$.
  - To the middle: $\Theta(n)$.

- Removing an element:
  - From the back: $\Theta(1)$.
  - From the front: $\Theta(n)$.
  - From the middle: $\Theta(n)$.

- Accessing an element:
  - At the back: $\Theta(1)$.
  - At the front: $\Theta(1)$.
  - At the middle: $\Theta(1)$.

Problem 9. Give the worst-case running times of the following operations of a doubly-linked-list-based (e.g., C++ STL list) list containing $n$ elements.
• Adding an element:
  – To the back.
  – To the front.
  – To the middle.

• Removing an element:
  – From the back.
  – From the front.
  – From the middle.

• Accessing an element:
  – At the back.
  – At the front.
  – In the middle.

Solution 9. Worst-case running times:

• Adding an element:
  – To the back: $\Theta(1)$.
  – To the front: $\Theta(1)$.
  – To the middle: $\Theta(n)$.

• Removing an element:
  – From the back: $\Theta(1)$.
  – From the front: $\Theta(1)$.
  – From the middle: $\Theta(1)$.

• Accessing an element:
  – At the back: $\Theta(1)$.
  – At the front: $\Theta(1)$.
  – In the middle: $\Theta(n)$.

Problem 10. Analyze the running time of the \texttt{union} function you wrote for Problem 6.

Problem 11. Analyze the running time of the \texttt{print_indices} function you wrote for Problem 7.