Data Structure : CSC 21200
Fall 2020
Prof. George Wolberg

Md Shahid Ershad
2372618

Midterm (Part 2 : Chap 4-9)

Chapter 4

Problem 4.1

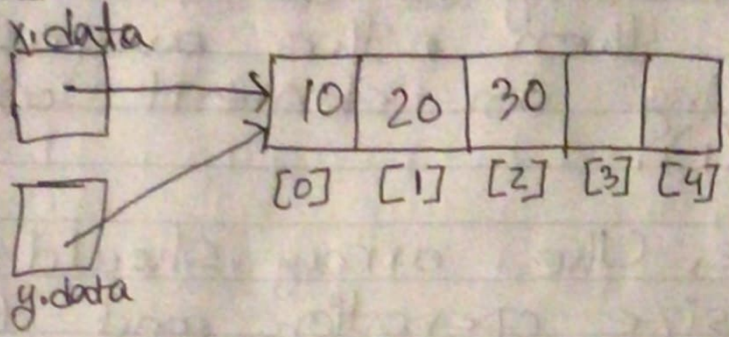
```
int *p = new int[100]; // *p is initialize to an array of 100 intergers.

for (int i=0; i<100; i++) { // this for loop will keep iterate from 0 to 99 and place them inside array
    P[i] = i;
}
delete[] p; // return the array to the heap.
```

Problem 4.2

As the constants are used, we will not be able to change the value of p in the function definition or implementation foo. Therefore, the pointer can not be modified.

Problem 4.3



As both pointers would point to the same dynamic array, any changes in x'array will have an impact ~~on~~ in y'array as well. If x'array is changed, then y'array would change as well.

Problem 4.4

As we know, dynamically ~~resizing~~ resizing is a costly operation.

```
reserved(used+1);
```

This is increasing the bag one by one. ~~Through~~ It would required to ~~constantly~~ repeat the insert operation and eventually it will take up a lot of or too much memory to do the process.

If we increase it by half its size instead of reducing the ~~memory~~ amount of memory used, the

[~~see~~ NEXT PAGE] →

If the resizing is done once every n operation then the average time for inserting an element will become O(1).

Therefore, the array should be doubled in size resize operation. and the correct way should be:

reserve(2*used);

Problem 4.5

D

Problem 4.6

A

Problem 4.7

A

Problem 4.8

B

Problem 4.9

B

Chapter 5

Problem 5.1

D

Problem 5.2

B

Problem 5.3

D

Problem 5.4

D

Problem 5.5

D

Problem 5.6

A

Chapter 6

Problem 6.1

When A template function is used, the compiler examines the data types of arguments and automatically determines the data type of ~~Hiern~~item. Therefore, we do not have to recompile files. It reduces the repetition of code.

Problem 6.2

The bug is it is replicating the values over and over again. It is overriding data. The value of data[0] is being copied throughout the entire array.

Correct code:

```
for(i=n; i>0; i--){
    data[i] = data[i-1];
}
```

Problem 6.3

The precondition and postcondition are given.

```
{
assert(head_ptr != NULL)
node<Item>* temp;
temp = head_ptr;
head_ptr = head_ptr -> link();
delete temp;
}
```

Problem 6.4

A

Problem 6.5

B

Problem 6.6

B

Problem 6.7

B

Chapter 7

Problem 7.1

C

Problem 7.2

A

Problem 7.3

D

Chapter 8

Problem 8.1

C

Problem 8.2

D

Chapter 9

Problem 9.1

```
#include <iostream>
using namespace std;
int print(int x){
 if (x<1){  // terminating condition
      return 0;
 }
 cout << " 1"; // printing "1"
 print (x-1); // calling the function.
 cout << "_"; // printing "_" (underscore)
```

Problem 9.2

The preconditions and postconditions are given.

Code:

```
#include <iostream>
int main() {
    int n, temp;  // calling variable
    cin >> n ;    // input n
    temp = n ;    // initializing temp = n
   cout << "The List is: " << endl
                  // outputing the list
   do {
     cout << n << endl ;
     n = n * 2 ;  // doubling n
   }
```

```
while (n < 4242) {};
    cout << n << endl;  // it will prin n
    do {
      cout << n << endl;
      n = n/2    // diving the n
    }
    while (n >= temp); // condition.
while (n >= temp); // condition
} //
// end of code.
```

Problem 9.3

D

Problem 9.4

D E