SCHOOL OF COMPUTER SCIENCES
UNIVERSITI SAINS MALAYSIA
Semester II Session 2023/2024

CPT113 – Programming Methodology & Data Structures
Tutorial Week 13
Recap

Learning Outcomes:
- Review the covered topics.

1- Given the following C++ program :

```
#include <iostream>
using namespace std;
main()
{

(1)  int *p, *q;
(2)  int x, y ,z;
(3)  x = 10; y = 15;
(4)  cout << &p << endl;
(5)  cout << &q << endl;
(6)  cout << &x << endl;
(7)  cout << &y << endl;
(8)  cout << x << endl;
(9)  cout << y << endl;
(10) p = &x;
(11) q = &y;
(12) *q = *p;
(13) cout << *p << endl;
(14) cout << *q << endl;
(15) cout << p << " " << q << endl;
(16) *p = 2 + *q;
(17) cout << x << endl;
(18) cout << y << endl;

}
```

a. Explain what each line (1 – 18) do before compiling the code.

b. Compile the code and check your answers with the output of the code.

2- Write C++program to do the following:

a. Declare a dynamic array named “first” type int with size 100;
b. Initialize the declared variable with odd numbers starting from 1;
c. Display all values in the array “first”
d. Declare a dynamic array named “second” as the same type and size as 2(a);

e. Copy the values of the array “first” to “second”.
f. Delete the array “first”
g. Display all values in the array “second”

3- Explain how this function `deleteNode` works?

```
void linkedListType::deleteNode(const int& deleteItem)
{
    nodeType *current;
    nodeType *trailCurrent;
    bool found;
    if(first == NULL)
       cout<<"Cannot delete from an empty list.\n";
     else
     {
        if(first->info == deleteItem)
        {
           current = first;
           first = first->link;
           count--;
           if(first == NULL)
              last = NULL;
           delete current;
        }
        else

      {
         found = false;
         trailCurrent = first;
         current = first->link;
         while(current != NULL && !found)
         {
            if(current->info != deleteItem)
            {
               trailCurrent = current;
               current = current->link;
            }
            else
               found = true;
         }
         if(found)
         {
            trailCurrent->link = current->link;
            count--;
            if(last == current)
               last = trailCurrent;
            delete current;
         }
         else
            cout<<”Item to be deleted is not in the list.”<<endl;
      } //end else
   } //end else
} //end deleteNode
```

4- What is the output of the following source-code?

```
int main()
{
   stack <int> intStack;
   queue <int> intQueue;
   int score[]={10,7,4,6,8};
   for (int I = 0; I < 5; i++)
   {
      intStack.push(score[i]);
      intQueue.push(score[i]);
   }
   intStack.pop();
   intQueue.push(intStack.top() - 5);
   intQueue.push(intStack.top());
   intQueue.pop();
   intStack.push(intQueue.front() + intStack.top());

   while (!intStack.empty())
   {
      cout << intStack.top() << "  ";
      intStack.pop();
   }

   cout << endl;

   while (!intQueue.empty())
   {
      cout << intQueue.front() << " ";
      intQueue.pop();
   }
}
```

5- Given the UML class diagram of the `stackType`

a. Write the class `stackType` definition.
b. Write all the functions definition in the class `stackType`
c. Write main () function to test the push(), pop() and top() functions.

stackType<Type>
-maxStackSize: int -stackTop: int -*list: Type
+operator=(const stackType<Type>&): const stackType<Type>& +initializeStack(): void +isEmptyStack() const: bool +isFullStack() const: bool +push(const Type&): void +top() const: Type +pop(): void -copyStack(const stackType<Type>&): void +stackType(int = 50) +stackType(const stackType<Type>&) +~stackType()

6- Consider the following recursive function:

```
int myRec(int x, int y)
{
   if (x == y)
      return x;
   else if (x > y)
      return (x + y);
   else
      return myRec(x + 1, y - 1);
}
```

a. State the base and general cases.

b. What is the output for the recursive calls myRec(5,10)? Show your working.

7- The sum of a series of consecutive numbers from 1 to n can be defined recursively as:

```
sum(1) = 1;
sum(n) = n+sum(n-1)
```

Write a recursive C++ function that accepts n as an argument and calculates the sum of the numbers from 1 to n.