2022.2 Object-Oriented Programming
Final Exam (December 16, 5pm-6:30pm)

supervisor signature	

StudentID# : ( ) , Name : ( )

*** You should write your answer in English. Otherwise, penalty may be applied.**

1. (18 points) Fill in the blanks (a)~(i) with the most appropriate English word(s).

(1) (a.) is a base class member function that you can redefine in a derived class to achieve polymorphism or dynamic binding.

(2) In STL, a container manages storage space for its elements and provides member functions to access them, either directly or through (b.). (same as (b)) provide common interface to step through the elements of any arbitrary type STL containers.

(3) (c.) means determining the exact implementation of a request based on both the request (operation) name and the receiving object at (d.) time. (same as (c)) allows us to (e.) new classes to existing systems without (f.) the existing code.

(4) Properties of abstract class :

- An abstract class contains at least one (g.).
- No (h.) of an abstract class can be created
- An abstract class can only be used as (i.).

2. (18 points) Consider following C++ code.

```
1: int x=5;
2: int* y;
3: int& z=x;
4: y=&x;
5: int w=*y;
6: x=9;
7: z=*y*x;
8: x++;
9: cout << x << " " << *y << " " << z << " " << w << endl;   // print the values
```

(1) What is the meaning of '*' in line 2 ? ( )

(2) What is the meaning of '&' in line 3 ? ( )

(3) What is the meaning of '&' in line 4 ? ( )

(4) What is the meaning of the first '*' and the second '*' in '*y*x' in line 7 ?
the meaning of the first '*': ( ), the meaning of the second '*': ( )

(5) What is the output result of executing line 9? ( )

3. (8 points) Fill in empty blanks with appropriate words.

(1) Container class is a holder object that stores ( )

(2) What is a priority queue? Explain with sufficient details. ( )

4. (12 points) There are differences among following three function parameter types (a), (b), and (c). `X` is a class name and `T` is a data type.

(a) `void X::f(T arg)`
(b) `void X::f(T& argp)`
(c) `void X::f(const T& argp)`

(1) What are the purposes of using '&' in (b) compared to using parameter type (a)? There are two important purposes (benefits). Explain with sufficient details.

(purpose 1:)
(purpose 2:)

(2) what is the purpose (benefit) of using `const` in (c) compared to using parameter type (b)? Explain.
( )

5. (6 points) Fill in the blanks below with the most appropriate word(s).
(1) What is the meaning of '**protected**' access (visibility) modifier? Explain.
( )

(2) Using '**protected**' instead of '**privatee**' means we have less ( ). Therefore, it is desirable to use '**protected**' only where it is really necessary.

6. (10 points) What is the output of following C++ code.

```
#include <iostream>
using namespace std;

class Address {
public:
    Address() { cout<< "1" << endl; }
    ~Address() { cout<< "2" << endl; }
};

class Person {
public:
    Person() { cout<< "3" << endl; }
    ~Person() { cout<< "4" << endl; }
};
```

```
class Student : public Person {
private:
    Address address;
public:
    Student() { cout<< "5" << endl; }
    ~Student() { cout<< "6" << endl; }
};

int main() { Student x; }
```

(write your answer here)
program output:

7. (14 points) C++ code below shows generic stack implementation using template. Fill in empty boxes with appropriate codes.

```
template<typename T>
class Stack {
    int size;
    int top;
    T *stackPtr;
public:
    Stack(int n) { size=n; top=0;   stackPtr=new T[size];  }
    ~Stack() { delete[] stackPtr; }
    bool push(                    );  // return true if push is successful
                                      // return false if the stack is full
    bool pop(                     );  // return true if pop is successful
                                      // return false if the stack is empty
    bool isEmpty() { if (top<=0) return true; else return false; }
    bool isFull() { if (top>=size) return true; else return false; }
};
```

```
// Insert your code for push member function
```

```
// Insert your code for pop member function
```

```
#include <iostream>
int main()
{
   int x, y;
   float xf, yf;
   Stack<int> s1(5);
   Stack<float> s2(5);
   s1.push(5); s1.push(8);
   s1.pop(x);  s1.pop(y);
   s2.push(5.3); s2.push(8.1);
   s2.pop(xf);  s2.pop(yf);
   std::cout <<x<< " " <<y<< std::endl;
   std::cout <<xf<< " " <<yf<< std::endl;

   return 0;
}
```

Output :

```
8 5
8.1 5.3
```

8. (14 points) Write a C++ function "**mySwap**" that takes two parameters **x** and **y**, and swaps the values of the two parameters (meaning it assigns the value of **x** to **y** and the value of **y** to **x**). Note that the types of **x** and **y** are the same but the type is a generic type. Therefore, **you must use template** to write the "**mySwap**" function that can accept any built-in type of parameters as shown in the following sample code and its output result.

```
#include <iostream>
int main()
{
    int a=3, b=4;
    float c=3.5 , d=2.3;
    mySwap(a,b);
    mySwap(c,d);
    std::cout << a << "," << b << "," << c << "," << d << "\n";
    return 0;
}
```

```
[output]
4,3,2.3,3.5
```

(Write your mySwap function here using template.)