National University of Computer and Emerging Sciences, Lahore Campus

Course:	Object Oriented Programming	Course Code:	CS 217
Program:	BS (CS, SE, DS)	Semester:	Spring 2021
Duration:	180 Min	Total Marks:	95
Paper Date:	5-Jul-2021	Page(s):	16
Section:	ALL	Section:	
Exam:	Final	Roll No:	

Roll Number: ________________________

Instructions:

Attempt all questions

You might use extra sheets for working but do not attach them.

Write the final answer in the space provided for it.

Questions are NOT ALLOWED during exam, take reasonable assumptions where needed.

Question No.	Marks	Marks Obtained
Q1	10	
Q2	15	
Q3	20	
Q4	25	
Q5	25	
Total	95	

Roll Number: ______________ **Section: _____________**

Question 1: (5+5 marks)

Part (A) Partial output of the code is given in right column, write the output in rows with question mark?

Code	Output
`#include "Person.h";// it’ s a fully functional class`	

`//template function`
`template <class T>`
`void my_swap(T &one, T &two)`
`{`
`   T temp = one;`
`   one= two;`
`   two= temp;`
`   cout<<"Swap successful";`
`}`	

`template <>`
`void my_swap (Person &one, Person &two)`
`{`
`      cout<<"You cannot swap Person ";`
`}`	

`int main()`	
`{`	
`   int a=10, b=20;`	
`   cout <<a<<" "<<b<<endl;`	10 20
`   my_swap(a,b);`	**Swap Successful**
`   cout <<a<<" "<<b<<endl;`	**20 10**
`   double *x= new double(10.5);`	
`   double *y= new double(11.5);`	
`   cout <<*x<<" "<<*y<<endl;`	10.5 11.5
`   my_swap(*x,*y);`	**Swap Successful**
`   cout <<*x<<" "<<*y<<endl;`	**11.5 10.5**
`   //overloaded constructor takes account name as input`	
`   Person P1("Ron"), Person P2("Harry");`	
`   cout<<P1<<" "<<P2;`	Ron Harry
`   my_swap(P1, P2);`	**You cannot swap person**

`   cout<<P1<<" "<P2;`
`}`
	Ron Harry

How many instances (copies) of my_swap functions are created at compile time in above code?	**3, one for int one for double and one for person**

Roll Number: _____________ **Section:** ____________

Part (B) The main function in code asks user to enter the type of transaction and then the amount, what will be the output of code given the user wants to enter following inputs.

Note: **`invalid_argument`** and **`out_of_range`** are subtypes of **`exception`** class.

```cpp
void getTransType(char &a)
{
  cout << "Enter W for withdraw and D for deposit"<<endl;
  cin>>a;
  if (a!='W' && a!='D')
    throw invalid_argument("Incorrect Transaction type");
}
void getAmount(int &amount)
{
  cout<<"Enter the amount";
  cin>>amount;

  if (amount<1)
    throw out_of_range("Enter positive number");
  if (amount>5000)
    throw exception("Amount should be less than 5001");
}
int main(){
  try{
      char a;
      getTransType(a);
      try{
            int amount;
            getAmount(amount);
            cout<<"Successful transaction";
      }
      catch(exception e){
            cout<<e.what();
      }
  }
  catch(invalid_argument ia) {
      cout<<ia.what();
  }
}
```

1) User enters ‘W’ and then 0 **Enter positive number**	**2)** User wants to enters ‘D’ and then 6000 **Amount should be less than 5001**	**3)** User want to enter ‘S’ and then 6000 **Incorrect Transaction type**

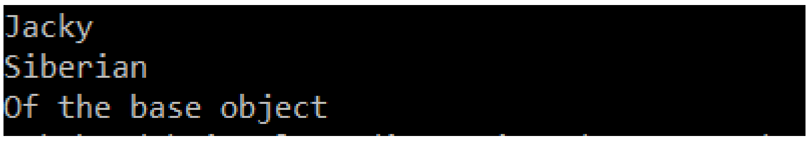
Question 2: (5+5+5 marks)

Write the output of the following code segments. Please note that all the code segments are error free.

Part (A)

```
class baseClass{
public:
	string name;
	string traits;
	double age;
};
class petClass {
private:
	string name;
	string traits;
	double * age; //age in years
public:
	baseClass baseObj;

	petClass(string name="Bailey", string traits="Husky", double* age=NULL){
		this->name=name;
		this->traits=traits;
		this->age= age;
	}

	petClass(baseClass baseObj){
		cout<<baseObj.name<<endl<<baseObj.traits<<endl<<"Of the base object";
	}

	void showPet(){
		cout << this->name<<"\t"<< this->traits;
		cout<<"\t" << *this->age << endl;
	}
};

void main(){
	petClass *dog= new petClass();
	dog->baseObj.name="Jacky";
	dog->baseObj.traits="Siberian";
	dog->baseObj.age= 4.5;
	petClass * myDog = new petClass(dog->baseObj);
	myDog->showPet();
	delete myDog;
	delete dog;
}
```

Output:



```
Jacky
Siberian
Of the base object
```

Roll Number: _____________ **Section: ____________**

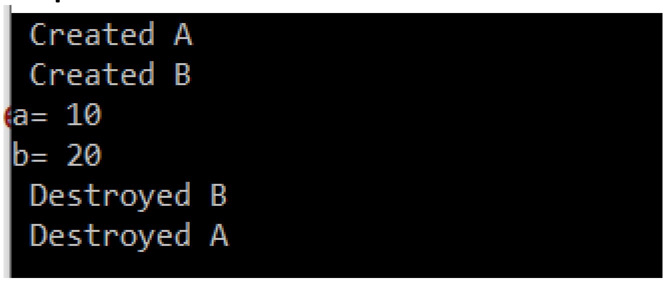
Part (B)

```
class A
{
	int a;
public:
	A(int a)
	{
		this->a=a;
		cout<<" Created A"<<endl;

	}
	virtual void print()
	{
		cout<<"a= "<<a<<endl;
	}

	virtual ~A()
	{
		cout<<" Destroyed A"<<endl;
	}
};
```

```
class B: public A {
	int b;

public:
	B(int a, int b):A(a){
		this->b=b;
		cout<<" Created B"<<endl;

	}
	void print()
	{
		A::print();
		cout<<"b= "<<b<<endl;
	}
	~B()
	{
		cout<<" Destroyed B"<<endl;
	}
};
```

```
int main()
{
	A *aPtr= new B(10, 20);
	aPtr->print();
	delete aPtr;

	return 0;
}
```

Output:



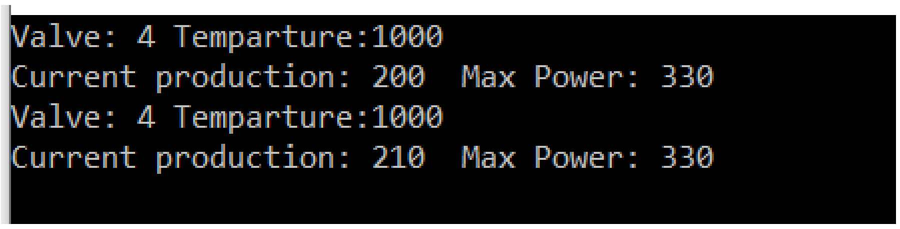
```
 Created A
 Created B
a= 10
b= 20
 Destroyed B
 Destroyed A
```

Part (C)

```
class ThermalReactor{
	int valve;
	float temprature;
public:
	ThermalReactor(int v, float t)
	{
		valve=v;
		temprature=t;
	}

	virtual void print(){
			cout<<"Valve: "<<valve;
			cout<<" Temparture:" <<temprature<<endl;
	}
};
```

```
class MagnoxReactor: public ThermalReactor
{
	float maxPower;
	float production;

public:
	MagnoxReactor(int v, float t, float m, float p)
	:ThermalReactor(v, t){
		maxPower=m;
		production=p;
	}

	bool isAtCritical(){ return(maxPower==production); }

	void signal(){ cout<<"Production cannot be increased"<<endl;}

	void increaseProd(float factor)
	{
		if((production+factor)<maxPower){
			production+=factor;
			print();
		}
		else signal();
	}

	void print(){
		ThermalReactor::print();
		cout<<"Current production: "<<production;
		cout<<"  Max Power: "<<maxPower<<endl;
	}
};

void Capacity(ThermalReactor * reactor ){
	reactor->print();
	dynamic_cast<MagnoxReactor *>(reactor)->increaseProd(10);

}

int main(){
	MagnoxReactor *MagRec= new MagnoxReactor(4, 1000, 330, 200);
	Capacity(MagRec); return 0;
}
```

Output:



```
Valve: 4 Temparture:1000
Current production: 200  Max Power: 330
Valve: 4 Temparture:1000
Current production: 210  Max Power: 330
```

Roll Number: _____________ **Section: ____________**

Question 3: (5+5+5+5 marks)

In each of the following code, indicate the error (**syntax, memory leakage**, **dangling pointer**)/output. Justify your answer(s).

`void main ()`
`{`
`    char name [] = "Abdur Rehman";`
`    char* const R = &name [6];`
`    const char* d = &name [2];`
`    cout << *(++R) << endl;`
`    cout << ++(*R) << endl;`
`    cout << *(++d) << endl;`
`    cout << ++(*d) << endl;`
`}`
	Syntax Error:
	In *(++R) → 'R': you cannot assign to a variable that is const
	In *(++d) → 'd': you cannot assign to a variable that is const	

`void alloc(int* a, int size) {`
`    a = new int[size];`
`}`
 	
`void main() {`
`    int* arr;`
`    alloc(arr, 10);`
`    arr[0] = 10;`
`}`
	memory leakage:
	Inside function, pointer a is passed by value, So, inside main, arr pointer does not have any memory after function call.	

`void allocate(int** a2d, int rows, int cols)`
`{`
`    a2d = new int* [rows];`
`    int** endptr = a2d + cols;`
`    for  (int  **temp=a2d;  temp<endptr;`
`temp++)`
`        temp = new int*[cols];`
`}`
`void main()`
`{`
`    int** x=nullptr;`
`    allocate(x, 3, 6);`
`}`
	memory leakage:
	Inside function, pointer a2d is passed by value, So, inside main, x pointer does not have any memory after function call.
	Also, inside allocate function, temp is being initialized with array of pointers, which is again total lost	

`int* sum(int* a) {`
`    int s = *a + *a;`
`    return &s;`
`}`
`void main() {`
`    int num = 10;`
`    int *sumPtr = sum(&num);`
`    cout << *sumPtr << endl;`
 	
`}`
	dangling pointer:
	Variable s inside the function is a local variable, which will be destroyed by the end of function execution. So, inside main sumPtr would be dangling.	

Roll Number: ______________ **Section: _____________**

Question 4: (25 marks)

We want to design a small calculator for calculation and storage of Chinese numbers from 0 to 10000. Following table represents the symbols and their corresponding pronunciation in English.

Arabic Numeral	0	1	2	3	4	5	6	7	8	9	10	100	1000	10000
Chinese Numeral	零 / 〇	一	二	三	四	五	六	七	八	九	十	百	千	万
Pronunciation	ling	yi	erh	san	ssu	wu	liu	chi	pa	chiu	shih	pai	chien	wan

The two-digit numbers are created by appending single digit number before them. Same rule applies to numbers with more digits. Examples with pronunciations are given below:

11	**12**	**13**	**20**	**21**	**30**	**309**	**9,008**	**1,111**	**99, 999**
shih yi	*shih erh*	*shih san*	*erh shih*	*erh shih yi*	*san shih*	*san pai chiu*	*chiu chien ling pa*	*yi chien yi pai yi shih yi*	*chiu wan chiu chien chiu pai chiu shih chiu*

These rules are followed when there is one or more 0's in a group of digits.

1) Do not pronounce the "numeral measure word" when it corresponds to 0

3,046 = *san chien ssu shih liu*

as 100 ***pai*** (百) corresponds to 0, there's no need to pronounce it.

2) Pronounce just one zero when there is more than one 0 in a group of 4 or more digits

3,007 = *san chien ling chi*

30,007 = *san wan ling chi*

30,107 = *san wan ling yi ling chi*

3) Do not pronounce 0 (or a group of 0) when it is at the end of a number (just ignore the 0）

8,000 = *pa chien*

55, 200 = wu wan wu chein erh

800 = pa pai, but 801 = pa pai yi

830 = pa pai san *chien*

Here is defined the class called **ChineseNum** to store positive Chinese numbers along with their equivalent English pronunciations. In this class the pronunciations of number are stored in a dynamic character 2D array, where each element of the array represents a single digit of the Chinese number.

```
class ChineseNum {
      //static arrays to store the digit names and positions from 0 to 10000.
      static char* positionNames[4];
      static char* digitNames[10];
      int num;            // Decimal number
      char** numName;     // Chinese digits Pronunciation
      int length;          //Number of digits
public:
      void encodeToChinese(); //Helper Function
};

char* ChineseNum::positionNames[4] = { "shih", "pai", "chien","wan" };
char* ChineseNum::digitNames[10]   = { "ling", "yi", "erh", "san" , "ssu", "wu",
                                        "liu", "chi", "pa", "chiu" };
```

A helper member function **encodeToChinese**() which encodes number from decimal to Chinses, without taking care of three rules.

```
void ChineseNum::encodeToChinese() {
	if (num<10) {
		length = 1;
		numName = new char* [length];
		numName[0] = new char[strlen(digitNames[num]) + 1];
		strcpy(numName[0], digitNames[num]);
	}
	else {
		int dcount = 0, n = num, d = 0;
		int digits[5] = { 0 };
		while (n) {
			digits[dcount++] = n % 10;
			n /= 10;
		}
		length = dcount + dcount - 1;
		numName = new char* [length];
		for (int i = 0, j = dcount - 1; i < length; i++, j--) {
			numName[i] = new char[strlen(digitNames[digits[j]]) + 1];
			strcpy(numName[i], digitNames[digits[j]]);
			i++;
			if (i < length) {
				numName[i] = new char[strlen(positionNames[j - 1]) + 1];
				strcpy(numName[i], positionNames[j - 1]);
			}
		}
	}
}
```

The above encoding function populates the numName as follows: This function does not take care of the three rules of zero's discussed above. You are not allowed to change this function, but you can use it where required.

Decimal Number (num)	2D array (numName)
5047	*wu chien ling pai ssu shih chi*
1	*yi*
5076	*wu chien ling pai chi shih liu*
90600	*chiu wan ling chien liu pai ling shih ling*

Now consider the following driver program. Add all the necessary methods to this class in order for the following driver program to work properly, without any compile, run time error or logical error. Specifically: provide the necessary **constructors** and **operators**. Read the comments in the driver program to get a hint about how the methods work.

```
void main() {
	ChineseNum n1(45);
	// Set decimal Number 45, and encodes
	ChineseNum n2; // by default the decimal value of a number is 0
	ChineseNum n4(90600);
	ChineseNum n3 = n1 + (++n2) + 5000;
	n1 = n3 + n2; // Sum the numbers and returns update number with Name
	cout << n1 << n2 << 30 + n3 << n4 << endl;
```

```
        // Prints the complete Number according to THE THREE RULES
        // Note: Rules are implemented only while printing, the original encoding saved
        in numName does not change,
        // you can use built in CString functions
}
```

The desired console output of main function is given below:

```
Number is: 5047  wu  chien  ling  ssu  shih  chi
Number is: 1  yi
Number is: 5076  wu  chien  ling  chi  shih  liu
Number is: 90600  chiu  wan  ling  liu  pai
```

Solution:

```
class ChineseNum {
        //static arrays to store the digit names from 0 to 10000.

        static char* positionNames[4];
        static char* digitNames[10];
        int num;                    // Decimal number
        char** numName;             // Chinese digits Names
        int length;  //Number of digits
public:
        ChineseNum();
        ChineseNum(int);
        ChineseNum(const ChineseNum&);
        ~ChineseNum();
        void encodeToChinese();
        void deallocateMemory();

        ChineseNum & operator=(ChineseNum&);
        ChineseNum  operator+(int);
        ChineseNum & operator++();
        ChineseNum  operator+(ChineseNum&);
        friend ChineseNum  operator+(int, ChineseNum&);
        friend ostream & operator<<(ostream&, const ChineseNum&);
};
char* ChineseNum::positionNames[4] = { "shih", "pai", "chien","wan" };
char* ChineseNum::digitNames[10] = { "ling", "yi", "erh", "san" , "ssu", "wu", "liu", "chi", "pa",
"chiu" };

ChineseNum::ChineseNum() {
        num = 0;
        encodeToChinese();
}

ChineseNum::ChineseNum(int num) {
        this->num = num;
        encodeToChinese();
}

ChineseNum::ChineseNum(const ChineseNum& n) {
        this->num = n.num;
        encodeToChinese();
}

ChineseNum::~ChineseNum() {
	deallocateMemory();
}

void ChineseNum::deallocateMemory() {
	for (int i = 0; i < length; i++) //deallocate whole memory
		delete[] numName[i];
	delete[] numName;
}

ChineseNum& ChineseNum::operator=(ChineseNum & n) {
	if (this != &n) {
		deallocateMemory();
		this->num = n.num;
		encodeToChinese();
	}
	return *this;
}

ChineseNum ChineseNum::operator+(int n) {
	ChineseNum r(this->num + n);
	r.encodeToChinese();
	return r;
}

ChineseNum ChineseNum::operator+(ChineseNum& n) {
	ChineseNum r(this->num + n.num);
	r.encodeToChinese();
	return r;
}

ChineseNum&  ChineseNum::operator++() {
	num++;
	deallocateMemory();
	encodeToChinese();
	return *this;
}

ChineseNum  operator+(int v, ChineseNum& n) {
	return n + v;
}


ostream& operator<<(ostream& out, const ChineseNum& n) {
	out << "Number is: " << n.num << "  ";

	if (n.length == 1) {
		out << n.numName[0] << "  ";
	}
	else {
		for (int i = 0; i < n.length; i++) {
			bool flag = false;
			// Rule No 1 do not add "shih", "pai", "chien","wan" corresponding to zeros
			if (i > 1 && strcmp(n.numName[i - 1], "ling") == 0)
			{
				for (int j = 0; j < 4; j++) {
					if (strcmp(n.numName[i], n.positionNames[j]) == 0)
						flag = true;
				}
			}
			// Rule No 2 Add single zero if there are consecutive zeros
			// Rule No 3 Exclude all trailing zeros
			else if (strcmp(n.numName[i], "ling") == 0)
			{
```

```
                                        if (i+2<n.length && strcmp(n.numName[i+2], "ling") == 0 ||
i==n.length-1)
                                                flag = true;
                        }
                        if (!flag)
                                out << n.numName[i] << "  ";
                }
        }
        out << endl;
        return out;
}

void ChineseNum::encodeToChinese() {
        if (num<10) {
                length = 1;
                numName = new char* [length];
                numName[0] = new char[strlen(digitNames[num]) + 1];
                strcpy(numName[0], digitNames[num]);
        }
        else {
                int dcount = 0, n = num, d = 0;
                int digits[5] = { 0 };
                while (n) {
                        digits[dcount++] = n % 10;
                        n /= 10;
                }
                length = dcount + dcount - 1;
                numName = new char* [length];
                for (int i = 0, j = dcount - 1; i < length; i++, j--) {
                        numName[i] = new char[strlen(digitNames[digits[j]]) + 1];
                        strcpy(numName[i], digitNames[digits[j]]);
                        i++;
                        if (i < length) {
                                numName[i] = new char[strlen(positionNames[j - 1]) + 1];
                                strcpy(numName[i], positionNames[j - 1]);
                        }
                }
        }
}
```

Roll Number: _____________ **Section: ____________**

Question 5: (25 marks)

We are required to develop a simple ***ChitChat*** application with following requirements. A **Chat** can have multiple Messages. Every **Message** has a RecivedDateTime *(DateTime)*. Our application supports two types of messages; **TextMessage** or **Sticker**. A **TextMessage** has some Text *(char*)* and a **Sticker** has a Filename*(char*)*.

Your task is to implement classes such that following main works successfully and produces the result given below. You are required to best use the Object-Oriented Programming concepts. Also, there shouldn't be any dangling pointer(s) or memory leakage in your code.

Note: Assume following functionality is already given. You may use it if required, do not re-write these functions.

```
void PrintSticker(char* fileName)
{
      //Assume that this function is already written
      //and it prints GIF according to the filename.
}

class DateTime
{
      int Day;
      int Month;
      int Year;
      int Hour;
      int Minutes;
public:
      DateTime(int day=0, int month=0, int year = 0, int hour=0, int mins = 0)
      {
            Day = day;
            Month = month;
            Year = year;
            Hour = hour;
            Minutes = mins;
      }
      void PrintTime()
      {
            cout<<Hour<<":"<<Minutes<<endl;
      }
      static DateTime GetCurrentDateTime()
      {
            // This function returns an object of Current DateTime
            // initialized with the values of Current System Date and Time
            time_t now = time(0);
            tm *ltm = localtime(&now);
            DateTime curr(ltm->tm_mday, 1 + ltm->tm_mon, 1900 + ltm->tm_year, ltm-
>tm_hour, ltm->tm_min);
            return curr;
      }
};
```

```
void main()
{
	Chat myChat(10);  //We can have total 10 messages at max.
	myChat.AddMessage(new TextMessage("Hello"));
	myChat.AddMessage(new TextMessage("Best of Luck for your Exams"));
	myChat.AddMessage(new Sticker("ThumbsUp.gif"));

	myChat.PrintAllMessages();
}
```

Required Output:

```
Hello.....10:34
Best of Luck for your Exams.....10:34
(Y).....10:34
```

Solution:

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

void PrintSticker(char* fileName){
	//Assume that this function is already written
	//and it prints GIF according to the filename.
	cout<<"(Y)";
}

class DateTime
{
	int Day;
	int Month;
	int Year;
	int Hour;
	int Minutes;
public:
	DateTime(int day=0, int month=0, int year = 0, int hour=0, int mins = 0){
		Day = day;
		Month = month;
		Year = year;
		Hour = hour;
		Minutes = mins;
	}
	void PrintTime()
	{
		cout<<Hour<<":"<<Minutes<<endl;
	}
	static DateTime GetCurrentDateTime()
	{
		// This function returns an object of Current DateTime
		// initialized with the values of Current System Date and Time
		time_t now = time(0);
		tm *ltm = localtime(&now);
		DateTime curr(ltm->tm_mday, 1 + ltm->tm_mon, 1900 + ltm->tm_year, ltm->tm_hour, ltm-
>tm_min);
		return curr;
	}
};


class Message
{
	DateTime RecievedDateTime;
public:
	Message()
	{
		RecievedDateTime = DateTime::GetCurrentDateTime();
	}
	virtual void Print()
	{
		cout<<".....";
		RecievedDateTime.PrintTime();
	}
	virtual ~Message()
	{
		//cout<<"~Message() Called.\n";
	}
};
class TextMessage: public Message
{
	char* Text;

public:
	TextMessage(char* text = "")
	{
		int str_len = strlen(text);
		Text = new char[str_len+1];
		for (int i = 0; i < str_len; i++)
		{
			Text[i] = text[i];
		}
		Text[str_len] = '\0';
	}
	void Print()
	{
		cout<<Text;
		Message::Print();
	}
	~TextMessage()
	{
		//cout<<"~TextMessage() Called.\n";
		if(Text != 0)
			delete[] Text;
	}
};
class Sticker : public Message
{
	char* fileName;
public:
	Sticker(char* _fileName)
	{
		int str_len = strlen(_fileName);
		fileName = new char[str_len+1];
		for (int i = 0; i < str_len; i++)
		{
			fileName[i] = _fileName[i];
		}
		fileName[str_len] = '\0';
	}
	void Print()
	{
		PrintSticker(fileName);
		Message::Print();
	}
	~Sticker()
	{
		//cout<<"~Sticker() Called.\n";
		if(fileName != 0)
		{
			delete[] fileName;
		}
	}
};
class Chat
{
	Message** Messages;
	int NoOfMessages;
	int maxLimit;
public:
	Chat(int n)
	{
		maxLimit = n;
		Messages = new Message*[n];
		NoOfMessages = 0;
	}
	void AddMessage(Message* msg)
	{
		Messages[NoOfMessages++] = msg;
	}
	void PrintAllMessages()
	{
		for(int i=0 ; i<NoOfMessages; i++)
			Messages[i]->Print();
	}
	~Chat()
	{
		//cout<<"~Chat() Called.\n";
		for(int i=0 ; i<NoOfMessages; i++)
			delete Messages[i];
		delete[] Messages;
	}
};

void main()
{
	Chat myChat(10);     //We can have total 10 messages at max.
	myChat.AddMessage(new TextMessage("Hello"));
	myChat.AddMessage(new TextMessage("Best of Luck for your Exams"));
	myChat.AddMessage(new Sticker("ThumbsUp.gif"));

	myChat.PrintAllMessages();
}
```