**Roll No…………….. Total No. of Pages:……**

**ST-2 (SET-III )**

**4th SEMESTER 2022-23**

**22CS006- Object Oriented Programming**

**Time allowed: 90 Minutes Max. Marks: 40**

**General Instructions:**

* **Follow the instructions given in each section.**
* **Make sure that you attempt the questions in order.**

**SECTION-A (10\*1 mark=10 marks)**

***(All questions are compulsory)***

1. Can a class inherit multiple classes in C++?
   1. Yes, but only up to three base classes
   2. Yes, but only if the base classes are in different namespaces
   3. No, C++ only supports single inheritance
   4. **Yes, C++ supports multiple inheritance**
2. What is the scope of a destructor in C++?
   1. Local to the class
   2. Global in the program
   3. **Limited to the object's lifetime**
   4. Limited to the class's lifetime
3. Can constructors be virtual in C++?
   1. Yes
   2. **No**
   3. Only if they are private
   4. Only if they are static
4. What is the purpose of a pure virtual function in C++?
   1. To prevent objects from being created from a class
   2. **To force derived classes to override the function**
   3. To deallocate memory before the object is destroyed
   4. To define the default behavior of a class member
5. What is the difference between an abstract class and an interface in C++?
   1. An abstract class cannot have constructors, while an interface can.
   2. An abstract class can have data members, while an interface cannot.
   3. **An abstract class can have both concrete and pure virtual functions, while an interface can only have pure virtual functions.**
   4. An abstract class can be inherited, while an interface cannot.
6. How is a destructor identified in C++?
   1. **By the '~' symbol followed by the class name**
   2. By the '+' symbol followed by the class name
   3. By the '^' symbol followed by the class name
   4. By the '-' symbol followed by the class name
7. When should you use a virtual destructor in a base class?
   1. When the base class has no derived classes
   2. **When the base class has at least one virtual function**
   3. When the base class is used to create objects
   4. When the base class is a singleton
8. What is the purpose of the 'this' pointer in C++?
   1. To access static class members
   2. **To access the object's address inside a member function**
   3. To call the base class constructor
   4. To deallocate memory for an object
9. Can a destructor be overloaded in C++?
   1. Yes
   2. **No**
   3. Only if the class has multiple constructors
   4. Only if the class has multiple member functions
10. What happens if a class has both a copy constructor and a copy assignment operator?
    1. The program will not compile.
    2. **The copy constructor will be called during assignment.**
    3. The copy assignment operator will be called during object creation.
    4. The compiler will randomly choose which one to call.

**SECTION-B (5\*2 mark=10 marks)**

***(All questions are compulsory)***

1. Which operator should be overloaded in the following code to make the program error free?

#include <iostream>

#include <string>

using namespace std;

class Box{

int capacity;

public:

Box(){}

Box(double capacity){

this->capacity = capacity;

}

};

int main(int argc, char const \*argv[])

{

Box b1(10);

Box b2 = Box(14);

if(b1 == b2){

cout<<"Equal";

}

else{

cout<<"Not Equal";

}

return 0;

}

a) +

**b) ==**

c) =

d) ()

1. What is the output of the following code?

#include <iostream>

using namespace std;

class A {

public:

A() { cout << "A constructor" << endl; }

~A() { cout << "A destructor" << endl; }

};

class B : public A {

public:

B() { cout << "B constructor" << endl; }

~B() { cout << "B destructor" << endl; }

};

class C : public B {

public:

C() { cout << "C constructor" << endl; }

~C() { cout << "C destructor" << endl; }

};

int main() {

C c;

return 0;

}

**a) A constructor, B constructor, C constructor, C destructor, B destructor, A destructor**

b) C constructor, B constructor, A constructor, A destructor, B destructor, C destructor

c) C constructor, B constructor, A constructor, A destructor, B destructor

d) A constructor, B constructor, C constructor, A destructor, B destructor, C destructor

1. When base class is derived in protected mode, then\_\_\_\_\_\_\_\_\_\_\_\_\_ .

1. public members of base class become private members of derived class.

2. public members of base class become protected members of derived class.

3. public members of base class become public members of derived class.

4. protected members of base class become protected members of derived class.

5. protected members of base class become private members of derived class.

6. protected members of base class become public members of derived class.

1. Only 1, 5
2. Only 1, 6
3. Only 2, 6
4. **Only 2, 4**
5. Give the function prototype of the operator function which we need to define in this program so that the program has no errors.

#include <iostream>

#include <string>

using namespace std;

class Box{

int capacity;

public:

Box(){}

Box(double capacity){

this->capacity = capacity;

}

};

int main(int argc, char const \*argv[])

{

Box b1(10);

Box b2 = Box(14);

if(b1 == b2){

cout<<"Equal";

}

else{

cout<<"Not Equal";

}

return 0;

}

1. bool operator==();
2. bool operator==(Box b){}
3. **bool operator==(Box b);**
4. Box operator==();
5. In case of inheritance where both base and derived class are having constructor and destructor, then which if the following are true ?

1. Constructors are executed in their order of derivation

2. Constructors are executed in reverse order of derivation

3. Destructors are executed in their order of derivation

4. Destructors are executed in reverse order of derivation

1. Only 2 ,4
2. Only 1 , 3
3. **Only 1 , 4**
4. Only 2, 3

**SECTION-C(Coding Question) (2x5 marks=5 marks)**

Q16) You have given a string. Write a function that reverses a string using a stack data structure.

**Input :** hello

**Output:** olleh

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | JAVA | Stack Overflow | Pune |
| **Output** | AVAJ | wolfrevO kcatS | enuP |

Solution :

**#include <stdio.h>**

**#include <string.h>**

**#define MAX\_SIZE 100**

**char stack[MAX\_SIZE];**

**int top = -1;**

**void push(char ch) {**

**//check for stack is full or not**

**if (top == MAX\_SIZE - 1) {**

**printf("Stack Overflow\n");**

**return;**

**}**

**//push element to stack**

**stack[++top] = ch;**

**}**

**char pop() {**

**//check for stack is empty or not**

**if (top == -1) {**

**printf("Stack Underflow\n");**

**return -1;**

**}**

**//remove element from top of the stack**

**return stack[top--];**

**}**

**void reverseString(char\* str) {**

**int len = strlen(str);**

**//push all characters of the string one by one**

**for (int i = 0; i < len; i++)**

**push(str[i]);**

**//pop all characters of the string one by one and store in same string**

**for (int i = 0; i < len; i++)**

**str[i] = pop();**

**//after this, str will contain reversed string**

**}**

**int main() {**

**char str[100];**

**printf("Enter a string: ");**

**scanf("%[^\n]%\*c", str);**

**printf("Original string: %s\n", str);**

**reverseString(str);**

**printf("Reversed string: %s\n", str);**

**return 0;**

**}**

Q17) You are tasked with developing a program that performs number swapping using macros.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | 5 10 | 18 93 | 9 7 |
| **Output** | Before swap: x = 5, y = 10  After swap: x = 10, y = 5 | Before swap: x = 18, y = 93  After swap: x = 93, y = 18 | Before swap: x = 9, y = 7  After swap: x = 7, y = 9 |

Solution :

**#include <stdio.h>**

**#define SWAP(a, b) do { \**

**int temp = (a); \**

**(a) = (b); \**

**(b) = temp; \**

**} while (0)**

**int main() {**

**int x = 5, y = 10;**

**printf("Before swap: x = %d, y = %d\n", x, y);**

**SWAP(x, y); //cal SWAP Macro**

**printf("After swap: x = %d, y = %d\n", x, y);**

**return 0;**

**}**

**SECTION-D (Coding Question)(1x10 mark=10 mark)**

Q18) You are given an array of integers nums, there is a sliding window of size k which is moving from the very left of

the array to the very right. You can only see the k numbers in the window. Each time the sliding window moves right by one position.

Return the max sliding window.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | [1,3,-1,-3,5,3,6,7], k = 3 | [2, 3, 0, -6, 4, 3], k = 3 | [2, 3, 0, -6, 4, 3], k = 2 |
| **Output** | [3,3,5,5,6,7] | [3,3,4,4] | [3,3,0,4,4] |

Solution :

**#include<stdio.h>**

**#include<stdlib.h>**

**#define MAX\_SIZE 100**

**int adjMatrix[MAX\_SIZE][MAX\_SIZE];**

**int visited[MAX\_SIZE];**

**struct queue**

**{**

**int size;**

**int f;**

**int r;**

**int\* arr;**

**};**

**//check if queue is empty**

**int isEmpty(struct queue \*q){**

**if(q->r==q->f){**

**return 1;**

**}**

**return 0;**

**}**

**//check if queue is full**

**int isFull(struct queue \*q){**

**if(q->r==q->size-1){**

**return 1;**

**}**

**return 0;**

**}**

**//add element in queue**

**void enqueue(struct queue \*q, int val){**

**if(isFull(q)){**

**printf("This Queue is full\n");**

**}**

**else{**

**q->r++;**

**q->arr[q->r] = val;**

**// printf("Enqued element: %d\n", val);**

**}**

**}**

**//remove element from queue**

**int dequeue(struct queue \*q){**

**int a = -1;**

**if(isEmpty(q)){**

**printf("This Queue is empty\n");**

**}**

**else{**

**q->f++;**

**a = q->arr[q->f];**

**}**

**return a;**

**}**

**void BFS(int start,int numVertices){**

**// Initializing Queue (Array Implementation)**

**struct queue q;**

**q.size = 400;**

**q.f = q.r = 0;**

**q.arr = (int\*) malloc(q.size\*sizeof(int));**

**// BFS Implementation**

**int node;**

**printf("%d ", start);**

**visited[start] = 1;**

**enqueue(&q, start); // Enqueue i for exploration**

**while (!isEmpty(&q))**

**{**

**int node = dequeue(&q);**

**for (int j = 0; j < numVertices; j++)**

**{**

**if(adjMatrix[node][j] ==1 && visited[j] == 0){**

**printf("%d ", j);**

**visited[j] = 1;**

**enqueue(&q, j);**

**}**

**}**

**}**

**}**

**int main(){**

**int numVertices;**

**printf("Enter the number of vertices: ");**

**scanf("%d", &numVertices);**

**printf("Enter the adjacency matrix:\n");**

**for (int i = 0; i < numVertices; i++) {**

**for (int j = 0; j < numVertices; j++) {**

**scanf("%d", &adjMatrix[i][j]);**

**}**

**}**

**// Initialize visited array**

**for (int i = 0; i < numVertices; i++) {**

**visited[i] = 0;**

**}**

**int startVertex;**

**printf("Enter the starting vertex: ");**

**scanf("%d", &startVertex);**

**BFS(startVertex, numVertices);**

**return 0;**

**}**