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

**ST-2 (SET-II)**

**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. Which keyword is used to create an instance of a class on the heap?
   1. **new**
   2. malloc
   3. create
   4. heap
2. What happens if a derived class has its own constructor, but the base class doesn't have a default constructor?
   1. It will result in a compile-time error.
   2. The program will crash at runtime.
   3. **The derived class constructor will implicitly call the base class's default constructor.**
   4. The derived class cannot have its own constructor.
3. When does dynamic binding (runtime polymorphism) occur in C++?
   1. During compile-time
   2. During linking
   3. **During runtime**
   4. During preprocessing
4. Which of the following operator cannot be overloaded?
5. +
6. **?:**
7. –
8. %
9. What is operator overloading in C++?
   1. Overriding the operator meaning by the user defined meaning for user defined data type
   2. Redefining the way operator works for user defined types
   3. Ability to provide the operators with some special meaning for user defined data type
   4. **All of the mentioned**
10. What is the function of the copy constructor in C++?
    1. It is used to create a copy of a derived class.
    2. It is used to create a copy of a base class.
    3. **It is used to create a copy of an object.**
    4. It is used to create a copy of a constructor.
11. Can a destructor be explicitly called for an object in C++?
    1. Yes
    2. **No**
    3. Only for dynamically allocated objects
    4. Only for statically allocated objects
12. Which operator is used to access the address of a class object?
    1. \*
    2. **&**
    3. -
    4. |
13. What is the purpose of a constructor with default arguments?
    1. It is used to provide multiple constructors for a class.
    2. It is used to deallocate memory.
    3. **It is used to create objects without specifying all constructor parameters.**
    4. It is used to initialize static class members.
14. What is the order of execution of constructors and destructors in a derived class?
    1. Destructor, Base Class Constructor, Derived Class Constructor
    2. **Base Class Constructor, Derived Class Constructor, Destructor**
    3. Derived Class Constructor, Base Class Constructor, Destructor
    4. Base Class Constructor, Destructor, Derived Class Constructor

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

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

1. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class complex

{

int i;

int j;

public:

complex(int a, int b)

{

i = a;

j = b;

}

complex operator+(complex c)

{

complex temp;

temp.i = this->i + c.i;

temp.j = this->j + c.j;

return temp;

}

void show(){

cout<<"Complex Number: "<<i<<" + i"<<j<<endl;

}

};

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

{

complex c1(1,2);

complex c2(3,4);

complex c3 = c1 + c2;

c3.show();

return 0;

}

* 1. 4 + i6
  2. 2 + i2
  3. **Error**
  4. Segmentation fault

1. Which of the followings are true about constructors?
   * 1. A class can have more than one constructor.
     2. They can be inherited.
     3. Their address can be referred.
     4. Constructors cannot be declared in protected section of the class.
     5. Constructors cannot return values.
     6. Only 1,2,4
     7. 1,2,4,5
     8. 1,3,5
     9. **1,4,5**
2. In case of operator overloading, operator function must be \_\_\_\_\_\_ .
3. Static member functions
4. Non- static member functions
5. Friend Functions
   * + 1. Only 2
       2. Only 1, 3
       3. **Only 2 , 3**
       4. All 1 , 2, 3
6. What is the output of the following code?

#include <iostream>

using namespace std;

class Base {

public:

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

virtual ~Base() { cout << "Base destructor" << endl; }

};

class Derived : public Base {

public:

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

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

};

int main() {

Base\* ptr = new Derived();

delete ptr;

return 0;

}

* + - 1. **Base constructor, Derived constructor, Derived destructor, Base destructor**
      2. Derived constructor, Base constructor, Base destructor, Derived destructor
      3. Base constructor, Derived constructor, Base destructor
      4. Derived constructor, Base destructor

1. Which of the following statement is true?
   * + - 1. In Procedural programming languages, all function calls are resolved at compile-time
         2. In Object Oriented programming languages, all function calls are resolved at compile-time
2. **I only**
3. II only
4. Both I and II
5. Neither I nor II

**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;**

**}**