2. How many runtime error messages associated with exception?
a) 2
b) 4
c) 5
d) infinite

Answer:c
Explanation:There are five runtime error messages associated with exceptions.They are no handler for the exception, Unexpected exception thrown, An exception can only be re-thrown in a handler,During stack unwinding, a destructor must handle its own exception, Out of memory.

3. Which block should be placed after try block?
a) catch
b) throw
c) a or b
d) none of the mentioned
Answer:c
Explanation:None.

4. What is the output of this program?

```
#include <iostream>

using namespace std;

int main()

{

   double a = 10, b = 5, res;

   char Operator = '/';

   try

   {

      if (b == 0)

         throw "Division by zero not allowed";

      res = a / b;

      cout << a << " / " << b << " = " << res;

   }

   catch(const char* Str)

   { cout << "\n Bad Operator: " << Str
   }

   return 0;
```

```
}
```

a) 10
b) 2
c) Bad Operator
d) 10 / 5 = 2

Answer:d
Explanation:In this program, We are dividing the two variables and printing the result. If any one of the operator is zero means, it will arise a exception.
Output:

```
$ g++
gex.cpp $
a.out
10 / 5 =2
```

5. What is the output of this program?

```
#include <iostream>

using namespace std;

int main()

{

   try

   {

      throw 1;

   }

   catch (int a)

   {

      cout << "exception number:  " << a << endl;

      return 0;

   }

   cout << "No exception " << endl;

   return 0;
}
```

a) No exception
b) exception number
c) exception number: 1
d) none of the mentioned

Answer:c
Explanation:If we caught a integer value means, there will be an exception, if it is not a integer, there will

not be a exception.
Output:

```
$ g++ gex1.cpp
$ a.out
exception number: 1
```

6. What is the output of this program?

```
#include <iostream>

using namespace std;

int main()

{

   int a = 10, b = 20, c = 30;

   float  d;

   try

   {

      if ((a - b) != 0)

      {

         d = c / (a - b);

         cout << d;

      }

      else

      {

         throw(a - b);

   }
   }
   catch (int i)

   {

      cout<<"[expand title=" "] Answer is infinite "<<i;

   }

}
```

a) 10
b) -3
c) 15
d) none of the mentioned

Answer:b
Explanation:We are manipulating the values, if there is any infinite value means, it will raise an exception.
Output:

```
$ g++ gex2.cpp
$ a.out
-3
```

7. What is the output of this program?

```
#include <iostream>

using namespace std;

void test(int x)

{

   try

   {

      if (x > 0)

         throw x;

      else

         throw 'x';

   }

   catch(int x)
{

      cout<<"integer:"<<x;

   }

   catch(char x)

   {

      cout << "character:" << x;

   }

}

int main()

{

   test(10);

   test(0);
```

```
    }
```

a) integer:10character:x
b) integer:10
c) character:x
d) none of the mentioned

Answer:a
Explanation:We are passing the integer and character and catching it by using multiple catch statement. Output:

```
$ g++ gex3.cpp
$ a.out
integer:10character:x
```

8. What is the output of this program?

```
    #include <iostream>

    using namespace std;

    void PrintSequence(int StopNum)

    {

       int Num;
       Num = 1;

       while (true)

       {

          if (Num >= StopNum)

             throw Num;

          cout << Num << endl;

          Num++;

       }

    }

    int main(void)

    {

       try

       {

          PrintSequence(2);

       }
```

```
        catch(int ExNum)
        {
            cout << "exception: " << ExNum << endl;
        }
        return 0;
    }
```

a) 1
b) exception: 2
c) 1
exception:
2
d) none of the mentioned

Answer:c
Explanation:In this program, We are printing one and raising a exception at 2.
Output:

```
$ g++ gex4.cpp
$ a.out
1
exception:
2
```

9. Pick out the correct Answer.
a) Exceptions are not suitable for critical points in code.
b) Exception are suitable for critical points in code.
c) Both a & b
d) none of the mentioned

Answer:a
Exceptions:If there is many number of exceptions in the program means, We have to use multiple catch statement and it is hard to keep track of the program.

10. Where does the exceptions are used?
a) To preserve the program
b) Exceptions are used when postconditions of a function cannot be satisfied.
c) Exceptions are used when postconditions of a function can be satisfied.
d) none of the mentioned

Answer:c
Explanation:None.

11. Which is used to handle the exceptions in c++?
a) catch handler
b) handler
c) exceptional handler
d) none of the mentioned

Answer:c
Explanation:None.

12. Which type of program is recommended to include in try block?
a) static memory allocation
b) dynamic memory allocation
c) const reference
d) pointer

Answer:b
Explanation:While during dynamic memory allocation, Your system may not have sufficient resources to handle it, So it is better to use it inside the try block.

13. Which statement is used to catch all types of exceptions?
a) catch()
b) catch(Test t)
c) catch(â€¦)
d) none of the mentioned

Answer:c
Explanation:This catch statement will catch all types of exceptions that arises in the program.

14. What is the output of this program?

```
#include <iostream>

using namespace std;

int main()

{

   int x = -1;

   try

   {

      if (x < 0)

      {

         throw x;

      }

      else

      {

         cout<<x;

      }

   }

   catch (int x )
```

```
    {
        cout << "Exception occurred: Thrown value is " << x << endl;
    }
    return 0;
}
```

a) -1
b) 0
c) Exception occurred: Thrown value is -1
d) error
Answer:c
Explanation:As the given value is -1 and according to the condition, We are arising an exception.
Output:
$ g++
etae.cpp $
a.out
Exception occurred: Thrown value is -1

15. What is the output of this program?

```
#include <iostream>
#include <typeinfo>
using namespace std;
class Polymorphic {virtual void Member(){}};
int main ()
{
    try
    {
        Polymorphic * pb = 0;
        typeid(*pb);
    }
    catch (exception& e)
    {
        cerr << "exception caught: " << e.what() << endl;
    }
    return 0;
```

```
}
```

a) exception caught: std::bad_typeid
b) exception caught: std::bad_alloc
c) exception caught: std::bad_cast
d) none of the mentioned

Answer:a

Explanation:In this program, We used a bad type id for the polymorphic operator, So it is arising an bad_typeid exception.
Output:
$ g++
etae.cpp $
a.out
exception caught: std::bad_typeid

16. What is the output of this program?

```
#include <iostream>

#include <exception>

using namespace std;

void myunexpected ()

{

   cout << "unexpected handler

   called\n"; throw;

}

void myfunction () throw (int,bad_exception)

{

   throw 'x';

}

int main (void)

{

   set_unexpected

   (myunexpected); try

   {

      myfunction();

   }
```

```
        catch (int)

        {
}
        catch (bad_exception be)

        {

            cout << "caught bad_exception\n";

        }

        catch (...)

        {

            cout << "caught other exception \n";

        }

        return 0;

    }
```

a) unexpected handler called
b) caught bad_exception
c) caught other exception
d) both a & b

Answer:d
Explanation:In this program, We are calling set_unexpected and myfunction, So it is printing the output as the given.
Output:

```
$ g++
etae.cpp $
a.out
unexpected handler called
caught bad_exception
```

17. What is the output of this program?

```
    #include <iostream>

    using namespace std;

    int main()

    {

        int x = -1;

        char *ptr;
        ptr = new char[256];

        try
```

```
        {
            if (x < 0)
            {
                throw x;
            }
            if (ptr == NULL)
            {
                throw " ptr is NULL ";
            }
        }
        catch (...)
        {
            cout << "Exception occurred: exiting "<< endl;
        }
        return 0;
    }
```

a) -1
b) ptr is NULL
c) Exception occured: exiting
d) none of the mentioned

Answer:c
Explanation:catch(â€¦) is used to catch all types of exceptions arising in the program.
Output:

```
$ g++
etea.cpp $
a.out
Exception occured: exiting
```

18. What is the output of this program?

```
    #include <iostream>
    #include <exception>
    using namespace std;
    void myunexpected ()
    {
```

```
    cout << "unexpected called\n";
    throw 0;
}
void myfunction () throw (int)
{
    throw 'x';
}
int main ()
{
    set_unexpected
    (myunexpected); try
    {
        myfunction();
    }
    catch (int)
    {
        cout << "caught int\n";
    }
    catch (...)
    {
        cout << "caught other exception\n";
    }
    return 0;
}
```

a) caught other exception
b) caught int
c) unexpected called
d) both b & d

Answer:d
Explanation:As we are calling set_unexpected (myunexpected) function, this is printing as unexpected called and because of operator compliance it is arising an exception.
Output:

```
$ g++
etea.cpp $
```

a.out
unexpected
called caught int

19. How to handle error in the destructor?
a) throwing
b) terminate
c) both a & b
d) none of the mentioned

Answer:b
Explanation:It will not throw an exception from the destructor but it will the process by using terminate() function.

20. What kind of exceptions are available in c++?
a) handled
b) unhandled
c) static
d) dynamic

Answer:b
Explanation:None.

1.What is the output of following program? #include <iostream>

```
1.  #include <exception>
2.  using namespace std;
3.  void myunexpected ()
4.  {
5.    cout << "unexpected handler called\n";
6.    throw;
7.  }
8.  void myfunction () throw (int,bad_exception)
9.  {
10.   throw 'x';
11. }
12. int main (void)
13. {
14.   set_unexpected (myunexpected);
15.   try
16.   {
17.     myfunction();
18.   }
19.   catch (int)
20.   {
21.     cout << "caught int\n";
22.   }
23.   catch (bad_exception be)
24.   {
25.     cout << "caught bad_exception\n";
26.   }
27.   catch (...)
28.   {
29.     cout << "caught other exception \n";
```

```
30.         }
31.         return 0;
32.     }
```

a) unexpected handler called
b) caught bad_exception
c) caught other exception
d) both a & b
Answer:d
Explanation:In this program, We are calling set_unexpected and myfunction, So it is printing the output as the given.
Output:

```
$ g++
etae.cpp $
a.out
unexpected handler called
caught bad_exception
```

2.. What is the output of this program?

```
1.  #include <iostream>
2.  using namespace std;
3.  int main()
4.  {
5.     int x = -1;
6.     char *ptr;
7.     ptr = new char[256];
8.     try
9.     {
10.       if (x < 0)
11.       {
12.          throw x;
13.       }
14.       if (ptr == NULL)
15.       {
16.          throw " ptr is NULL ";
17.       }
18.    }
19.    catch (...)
20.    {
21.       cout << "Exception occurred: exiting "<< endl;
22.    }
23.    return 0;
24. }
```

a) -1
b) ptr is NULL
c) Exception occured: exiting
d) none of the mentioned
Answer:c
Explanation:catch(…) is used to catch all types of exceptions arising in the program.
Output:

```
$           g++
etea.cpp      $
a.out
Exception occured: exiting
```

3. What is the output of this program?

```
1.  #include <iostream>
2.  #include <exception>
3.  using namespace std;
```

```
4.  void myunexpected ()
5.  {
6.     cout << "unexpected called\n";
7.     throw 0;
8.  }
9.  void myfunction () throw (int)
10. {
11.    throw 'x';
12. }
13. int main ()
14. {
15.    set_unexpected (myunexpected);
16.    try
17.    {
18.       myfunction();
19.    }
20.    catch (int)
21.    {
22.       cout << "caught int\n";
23.    }
24.    catch (...)
25.    {
26.       cout << "caught other exception\n";
27.    }
28.    return 0;
29. }
```

a) caught other exception
b) caught int
c) unexpected called
d) both b &
d Answer:d
Explanation:As we are calling set_unexpected (myunexpected) function, this is printing as unexpected called and because of operator compliance it is arising an exception.
Output:

```
$ g++
etea.cpp $
a.out
unexpected
called caught int
```

4. What is the output of this program?

```
1.  #include <iostream>
2.  #include <typeinfo>
3.  using namespace std;
4.  class Polymorphic {virtual void Member(){}};
5.  int main ()
6.  {
7.     try
8.     {
9.        Polymorphic * pb = 0;
10.       typeid(*pb);
11.    }
12.    catch (exception& e)
13.    {
14.       cerr << "exception caught: " << e.what() << endl;
15.    }
16.    return 0;
```

```
    }
```
a) exception caught: std::bad_typeid
b) exception caught: std::bad_alloc
c) exception caught: std::bad_cast
d) none of the mentioned

Answer:a

Explanation:In this program, We used a bad type id for the polymorphic operator, So it is arising an bad_typeid exception.

Output:

$ g++
etae.cpp $
a.out
exception caught: std::bad_typeid

5.To where does the program control transfers when exception is arised?
a) catch
b) handlers
c) throw
d) none of the mentioned

Answer:b

Explanation:When a exception is arised mean, the exception is caught by handlers and then it decides the type of exception.

6. Which key word is used to check exception in the block of code?
a) catch
b) throw
c) try
d) none of the mentioned

Answer:c

Explanation:The try() statement is used for exceptios in c++.

7. What will happen when the exception is not caught in the program?
a) error
b) program will execute
c) block of that code will not execute d) none of the mentioned

Answer:a

Explanation:None.

8. What is the output of this program?

```
1.  #include <iostream>
2.  using namespace std;
3.  int main()
4.  {
5.      int age = 0;
6.      try {
7.          if (age < 0) {
8.              throw "Positive Number Required";
9.          }
10.         cout << age;
11.     }
12.     catch(const char *Message)
13.     {
14.         cout << "Error: " << Message;
15.     }
16.     return 0;
17. }
```

a) 0
b) Error:Positive Number Required

c) compile time error

d) none of the mentioned

Answer:a

Explanation:As the zero marks the beginning of the positive number, it is printed as output Output:

```
$               g++
excep.cpp       $
a.out
0
```

9. What is the output of this program?

```
1.  #include <iostream>
2.  using namespace std;
3.  void PrintSequence(int StopNum)
4.  {
5.     int Num;
6.     Num = 1;
7.     while (true) {
8.        if (Num >= StopNum)
9.           throw Num;
10.       cout << Num;
11.       Num++;
12.    }
13. }
14. int main(void)
15. {
16.    try {
17.       PrintSequence(20);
18.    }
19.    catch(int ExNum)
20.    {
21.       cout << "Caught an exception with value: " << ExNum;
22.    }
23.    return 0;
24. }
```

a) compile time error

b) prints first 19 numbers

c) prints first 19 numbers and throws exception at 20

d) none of the mentioned

Answer:c

Explanation:In this program, we are printing upto 19 numbers and when executing the 20, we are raising a exception.

Output:

```
$ g++ excep1.cpp
$ a.out
12345678910111213141516171819Caught an exception with value: 20
```

10. What is the output of this program?

```
1.  #include <iostream>
2.  using namespace std;
3.  double division(int a, int b)
4.  {
5.     if (b == 0) {
6.        throw "Division by zero condition!";
7.     }
8.     return (a / b);
9.  }
10. int main ()
```

```
11.     {
12.        int x = 50;
13.        int y = 2;
14.        double z = 0;
15.        try {
16.           z = division(x, y);
17.           cout << z;
18.        }
19.        catch(const char *msg) {
20.           cerr << msg;
21.        }
22.        return 0;
23.     }
```

a) 25
b) 20
c) Division by zero condition!
d) none of the mentioned
Answer:a

Explanation:In this program, we resembling the division by using the exception handling. Output:

```
$ g++ excep2.cpp
$ a.out
25
```

11. What is the output of this program?

```
1.     #include <iostream>
2.     using namespace std;
3.     int main()
4.     {
5.        char* buff;
6.        try {
7.           buff = new char[1024];
8.           if (buff == 0)
9.              throw "Memory allocation failure!";
10.          else
11.             cout << sizeof(buff) << "Byte successfully allocated!"<<endl;
12.       }
13.       catch(char *strg) {
14.          cout<<"Exception raised: "<<strg<<endl;
15.       }
16.       return 0;
17.    }
```

a) 4 Bytes allocated successfully
b) 8 Bytes allocated successfully
c) Memory allocation failure
d) depends on the size of data type
Answer:d

Explanation:As we are allocating the memory to the variables and if there is not suffcient size means, it will throw an exception.
Output:

```
$ g++ excep3.cpp
$ a.out
4 Bytes allocated successfully
```

12. What is the output of this program?

```
1.     #include <iostream>
2.     using namespace std;
```

```
3.  void Funct();
4.  int main()
5.  {
6.     try {
7.        Funct();
8.     }
9.     catch(double) {
10.       cerr << "caught a double type..." << endl;
11.    }
12.    return 0;
13. }
14. void Funct()
15. {
16.    throw 3;
17. }
```

a) caught a double type
b) compile time error
c) abnormal program termination
d) none of the mentioned

Answer:c

Explanation:As we are throwing integer to double it will raise as abnormal program after termination throw statement.

Output:

```
$ g++ excep4.cpp
$ a.out
terminate called after throwing an instance of 'int'
Aborted
```

13. What is the output of this program?

```
1.  #include <iostream>
2.  #include <exception>
3.  using namespace std;
4.  int main()
5.  {
6.     try {
7.        int * array1 = new int[100000000];
8.        int * array2 = new int[100000000];
9.        int * array3 = new int[100000000];
10.       int * array4 = new int[100000000];
11.       cout << "Allocated successfully";
12.    }
13.    catch(bad_alloc&) {
14.       cout << "Error allocating the requested memory." << endl;
15.    }
16.    return 0;
17. }
```

a) Allocated successfully
b) error allocating the requested memory
c) Depends on the memory of the computer
d) none of the mentioned

Answer:c

Explanation:In this program, we allocating the memory to the arrays by using excetion handling and we handled the exception by standard exception.

Output:

```
$ g++
excep5.cpp $
a.out
```

Allocated successfully

14.What is the output of this program?

```
1.  #include <iostream>
2.  #include <exception>
3.  using namespace std;
4.  class myexception: public exception
5.  {
6.     virtual const char* what() const throw()
7.     {
8.        return "exception arised";
9.     }
10. } myex;
11. int main ()
12. {
13.    try
14.    {
15.       throw myex;
16.    }
17.    catch (exception& e)
18.    {
19.       cout << e.what() << endl;
20.    }
21.    return 0;
22. }
```

a) exception arised

b) error

c) exception

d) runtime error

Answer:a

Explanation:In this program, We are arising a standard exception and catching that and returning a statement.

Output:

```
$ g++ goe.cpp
$ a.out
exception
arised
```

15. What is the output of this program?

```
1.  #include <iostream>
2.  using namespace std;
3.  int main()
4.  {
5.     int age=5;
6.     try
7.     {
8.        if (age < 0)
9.           throw "Positive Number Required";
10.       cout << age << "\n\n";
11.    }
12.    catch(const char* Message)
13.    {
14.     cout << "Error: " << Message;
15.    }
16.    return 0;
17. }
```

a) 5

b) 10
c) 15
d) Positive Number Required
Answer:a
Explanation:In this program, We are checking the age of a person, If it is zero means, We will arise a exception.
Output:

```
$ g++ goe1.cpp
$ a.out
5
```

16. What is the output of this program?

```
1.  #include <iostream>
2.  using namespace std;
3.  double division(int a, int b)
4.  {
5.     if (b == 0)
6.     {
7.        throw "Division by zero condition!";
8.     }
9.     return (a/b);
10. }
11. int main ()
12. {
13.    int x = 50;
14.    int y = 0;
15.    double z = 0;
16.    try
17.    {
18.       z = division(x, y);
19.       cout << z << endl;
20.    }
21.    catch (const char* msg)
22.    {
23.       cout << msg << endl;
24.    }
25.    return 0;
26. }
```

a) 50
b) 0
c) Division by zero condition!
d) none of the mentioned
Answer:c
Explanation:We are dividing the values and if one of the values is zero means, We are arising an exception.
Output:

```
$ g++ goe2.cpp
$ a.out
Division by zero condition!
```

17. What is the output of this program?

```
1.  #include <iostream>
2.  #include <string>
3.  using namespace std;
4.  int main()
5.  {
6.     double Op1=10, Op2=5, Res;
7.     char Op;
```

```
8.          try
9.          {
10.            if (Op != '+' && Op != '-' &&
11.               Op != '*' && Op != '/')
12.            throw Op;
13.            switch (Op)
14.            {
15.            case '+':
16.               Res = Op1 + Op2;
17.               break;
18.            case '-':
19.               Res = Op1 - Op2;
20.               break;
21.            case '*':
22.               Res = Op1 * Op2;
23.               break;
24.            case '/':
25.               Res = Op1 / Op2;
26.               break;
27.            }
28.            cout << "\n" << Op1 << " " << Op << " "<< Op2 << " = " << Res;
29.             }
30.             catch (const char n)
31.         {
32.            cout << n << " is not a valid operator";
33.         }
34.         return 0;
35.      }
```

a) 15
b) 5
c) 2
d) 1 is not a valid operator

Answer:d

Explanation:It will arise a exception because we missed a operator.

Output:

```
$ g++ goe3.cpp
$ a.out
1 is not a valid operator
```

18. What is the output of this program?

```
1.       #include <iostream>
2.       #include "math.h"
3.       using namespace std;
4.       double MySqrt(double d)
5.       {
6.          if (d < 0.0)
7.             throw "Cannot take sqrt of negative number";
8.          return sqrt(d);
9.       }
10.      int main()
11.      {
12.         double d = 5;
13.         cout << MySqrt(d) << endl;
14.      }
```

a) 5
b) 2.236
c) error

d) Cannot take sqrt of negative number

Answer:b

Explanation:We are finding the square root of the number, if it is a positive number, it can manipulate, Otherwise it will arise a exception.

Output:

```
$ g++ goe4.cpp
$ a.out
2.236
```

19.What is the output of this program?

```
1.  #include <iostream>
2.  using namespace std;
3.  void empty() throw()
4.  {
5.     cout << "In empty()";
6.  }
7.  void with_type() throw(int)
8.  {
9.     cout << "Will throw an int";
10.    throw(1);
11. }
12. int main()
13. {
14.    try
15.    {
16.       empty();
17.       with_type();
18.    }
19.    catch (int) {
20.       cout << "Caught an int";
21.    }
22. }
```

a) In empty()

b) Will throw an int

c) Caught an int

d) All of the mentioned

Answer:d

Explanation:It will print all three because we are calling all functions in the main(). Output:

```
$ g++ exs.cpp $
a.out
In empty()Will throw an intCaught an int
```

20. What is the output of this program?

```
1.  #include <iostream>
2.  #include <exception>
3.  #include <typeinfo>
4.  using namespace std;
5.  class Test1
6.  {
7.     virtual int  Funct()
8.     {
9.     }
10. };
11. int main ()
12. {
13.    try
```

```
14.         {
15.            Test1 * var = NULL;
16.            typeid (*var);
17.         }
18.         catch (std::exception& typevar)
19.         {
20.            cout << "Exception: " << typevar.what() << endl;
21.         }
22.         return 0;
23.      }
```

a) NULL
b) Exception:bad_alloc
c) Exception:std:bad_typeid
d) none of the mentioned
Answer:c
Explanation:As we are using a bad type on pointers, So it is arising an error.
Output:

```
$ g++ exs1.cpp
$ a.out
Exception:std:bad_typeid
```

21. What is the output of this program?

```
1.    #include <iostream>
#include <string>
2.    #include<typeinfo>
3.    using namespace std;
4.    int main( )
5.    {
6.       try
7.       {
9.          string strg1("Test");
10.         string strg2("ing");
11.         strg1.append(strg2, 4, 2);
12.         cout << strg1 << endl;
13.      }
14.      catch (exception &e)
15.      {
16.         cout << "Caught: " << e.what() << endl;
17.         cout << "Type: " << typeid(e).name() << endl;
18.      };
19.      return 0;
20.   }
```

a) out of range
b) bad type_id
c) bad allocation
d) none of the mentioAnswer:a
Explanation:As we are using out of bound value on strings, So it arising an exception. Output:

```
$ g++ exs2.cpp
$ a.out
Caught: basic_string::append
Type: St12out_of_range
#include
ned
```

1).What is the result of executing the following code, using the parameters 4 and 0:
public void divide(int a, int b) {

```
try {
int c = a / b;
} catch (Exception e) {
System.out.print("Exception
"); } finally {
System.out.println("Finally");
}
```

1) Prints out: Exception Finally
2) Prints out: Finally
3) Prints out: Exception
4) No output

Answer : 1

2).What is the o/p of the program?

```
try
{
   int x = 0;
   int y = 5 / x;
}
catch (Exception e)
{
   System.out.println("Exception");
}
catch (ArithmeticException ae)
{
   System.out.println(" Arithmetic Exception");
}
System.out.println("finished");
```

A.finished
B.Exception
C.Compilation fails.
D.Arithmetic Exception

Ans:-c

3). What is the o/p of the program?

```
public class X
{
   public static void main(String [] args)
   {
      try
      {
         badMethod();
         System.out.print("A");
      }
      catch (Exception ex)
      {
         System.out.print("B");
      }
      finally
      {
         System.out.print("C");
      }
      System.out.print("D");
   }
   public static void badMethod()
   {
```

```
        throw new Error(); /* Line 22 */
    }
}
```

A.ABCD
B.Compilation fails.
C.C is printed before exiting with an error message.
D.BC is printed before exiting with an error message. Ans:-c

4). What is the o/p of the program?.

```
public class X
{
    public static void main(String [] args)

{
        try
        {
            badMethod();
            System.out.print("A");
        }
        catch (RuntimeException ex) /* Line 10 */
        {
            System.out.print("B");
        }
        catch (Exception ex1)
        {
            System.out.print("C");
        }
        finally
        {
            System.out.print("D");
        }
        System.out.print("E");
    }
    public static void badMethod()
    {
        throw new RuntimeException();
    }
}
```

A.BD
B.BCD
C.BDE
D.BCDE
Ans:-c

5).What is the o/p of the program?

```
public class RTExcept
{
    public static void throwit ()
    {
        System.out.print("throwit ");
        throw new RuntimeException();
    }
    public static void main(String [] args)
    {
        try
        {
            System.out.print("hello ");
```

```
        throwit();
      }
      catch (Exception re )
      {
        System.out.print("caught ");
      }
      finally
      {
        System.out.print("finally ");
      }
      System.out.println("after ");
    }
}
```

A.hello throwit
caught
B.Compilation fails
C.hello throwit *RuntimeException* caught
after
D.hello throwit caught finally after
Ans:-d

6. What is the output of this program?

```
1.  class exception_handling {
2.      public static void main(String args[]) {
3.          try {
4.              System.out.print("Hello" + " " + 1 / 0);
5.          }
6.          catch(ArithmeticException e) {
7.              System.out.print("World");
8.          }
9.      }
10. }
```

a) Hello
b) World
c) HelloWorld
d) Hello
World ans:-b

7). What is the output of this program?

```
1.  class exception_handling {
2.      public static void main(String args[]) {
3.          try {
4.              int a, b;
5.              b = 0;
6.              a = 5 / b;
7.              System.out.print("A");
8.          }
9.          catch(ArithmeticException e) {
10.             System.out.print("B");
11.         }
12.     }
13. }
```

a) A
b) B
c) Compilation Error
d) Runtime

Error ans:-b

8). What is the output of this program?

```
1.  class exception_handling {
2.      public static void main(String args[]) {
3.          try {
4.              int a, b;
5.              b = 0;
6.              a = 5 / b;
7.              System.out.print("A");
8.          }
9.          catch(ArithmeticException e) {
10.             System.out.print("B");
11.         }
12.         finally {
13.                 System.out.print("C");
14.         }
15.     }
16. }
```

a) A
b) B
c) AC
d) BC
ans:-d

9).What is the output of this program?

```
1.  class exception_handling {
2.      public static void main(String args[]) {
3.          try {
4.              int i, sum;
5.              sum = 10;
6.              for (i = -1; i < 3 ;++i)
7.                  sum = (sum / i);
8.          }
9.          catch(ArithmeticException e) {
10.             System.out.print("0");
11.         }
12.         System.out.print(sum);
13.     }
14. }
```

a) 0
b) 05
c) Compilation Error
d) Runtime
Error ans:-b

10). What is the output of this program?

```
1.  class exception_handling {
2.      public static void main(String args[]) {
3.          try {
4.              int i, sum;
5.              sum = 10;
6.              for (i = -1; i < 3 ;++i) {
7.                  sum = (sum / i);
8.              System.out.print(i);
```

```
9.                  }
10.               }
11.               catch(ArithmeticException e) {
12.                  System.out.print("0");
13.               }
14.            }
15.         }
```

a) -1
b) 0
c) -10
d) -101
ans:-c

11).

What is the output of following try
catch block try
{

```
i
n
t

i
;

r
e
t
u
r
n
;
```

}
catch(Exception e)
{
System.out.println(“Hello India”);
}
finally
{
System.out.println(“Hello Morbi”);
}

A. Hello India

B. Hello India
Hello Morbi

C. Hello Morbi

D. the program will return without printing anything

Ans:-c

12). What is the output of this program?

```
1.       #include <iostream>
2.       using namespace std;
3.       int main()
4.       {
5.          int x = -1;
6.          try
7.          {
8.             if (x < 0)
9.             {
10.               throw x;
```

```
11.             }
12.             else
13.             {
14.                cout<<x;
15.             }
16.          }
17.          catch (int x )
18.          {


19.             cout << "Exception occurred: Thrown value is " << x << endl;
20.          }
21.          return 0;
22.       }
```

a) -1
b) 0
c) Exception occurred: Thrown value is -1
d) error
ans:-c

13). What is the output of this program?

```
1.       #include <iostream>
2.       #include <typeinfo>
3.       using namespace std;
4.       class Polymorphic {virtual void Member(){}};
5.       int main ()
6.       {
7.          try
8.          {
9.             Polymorphic * pb = 0;
10.            typeid(*pb);
11.         }
12.         catch (exception& e)
13.         {
14.            cerr << "exception caught: " << e.what() << endl;
15.         }
16.         return 0;
17.      }
```

a) exception caught: std::bad_typeid
b) exception caught: std::bad_alloc
c) exception caught: std::bad_cast
d) none of the mentioned
ans:-a

14). What is the output of this program?

```
1.       #include <iostream>
2.       #include <exception>
3.       using namespace std;
4.       void myunexpected ()
5.       {
6.          cout << "unexpected handler called\n";
7.          throw;
8.       }
9.       void myfunction () throw (int,bad_exception)
10.      {
11.         throw 'x';
```

```
12.     }
13.     int main (void)
14.     {
15.        set_unexpected (myunexpected);
16.        try
17.        {
18.           myfunction();
19.        }
20.        catch (int)
21.        {
22.           cout << "caught int\n";
23.        }
24.        catch (bad_exception be)
25.        {
26.           cout << "caught bad_exception\n";
27.        }
28.        catch (...)
29.        {
30.           cout << "caught other exception \n";
31.        }
32.        return 0;
33.     }
```

a) unexpected handler called
b) caught bad_exception
c) caught other exception
d) both a & b
ans:-d

15). What is the output of this program?

```
1.     #include <iostream>
2.     using namespace std;
3.     int main()
4.     {
5.        int x = -1;
6.        char *ptr;
7.        ptr = new char[256];
8.        try
9.        {
10.          if (x < 0)
11.          {
12.             throw x;
13.          }
14.          if (ptr == NULL)
15.          {
16.             throw " ptr is NULL ";
17.          }
18.       }
19.       catch (...)
20.       {
21.          cout << "Exception occurred: exiting "<< endl;
22.       }
23.       return 0;
24.    }
```

a) -1
b) ptr is NULL
c) Exception occured: exiting

c) none of the mentioned
ans:-c

16). What is the output of this program?

```
1.  #include <iostream>
2.  #include <exception>
3.  using namespace std;
4.  void myunexpected ()
5.  {
6.     cout << "unexpected called\n";
7.     throw 0;
8.  }
9.  void myfunction () throw (int)
10. {
11.    throw 'x';
12. }
13. int main ()
14. {
15.    set_unexpected (myunexpected);
16.    try
17.    {
18.       myfunction();
19.    }
20.    catch (int)
21.    {
22.       cout << "caught int\n";
23.    }
24.    catch (...)
25.    {
26.       cout << "caught other exception\n";
27.    }
28.    return 0;
29. }
```

a) caught other exception
b) caught int
c) unexpected called
d) both b & d
ans:-d

17). What will be the output of this code?

```
int var=12;
try{
   cout<<"Try block Entered"<<endl;
   if(var>30)
     throw var;
   cout<<"Try block Leaving"<<endl;
}catch(int thrown_value){
  cout<<"Exception value:"<<thrown_value<<endl;
}
cout<<"After try block"<<endl;
```

18). What will be the output of this code?

```
int var=50;
try{
   cout<<"Try block Entered"<<endl;
   if(var>30)
     throw var;
```

```
      cout<<"Try block Leaving"<<endl;
   }catch(int thrown_value){
     cout<<"Exception value:"<<thrown_value<<endl;
   }
   cout<<"After try block"<<endl;
```

19). What will be the output of this code?

```
int main()
{
  int x = -1;

  // Some code
  cout << "Before try
  \n"; try {
     cout << "Inside try
     \n"; if (x < 0)
     {
        throw x;
        cout << "After throw (Never executed) \n";
     }
  }
  catch (int x ) {
     cout << "Exception Caught \n";
  }

  cout << "After catch (Will be executed) \n";
  return 0;
}
```

Output:
Before try
Inside try
Exception Caught
After catch (Will be executed)

20).what is the o/p of
program?. #include <iostream>
using namespace std;

```
int main()
{
   try {
      throw
      'a';
   }
   catch (int x) {
       cout << "Caught ";
   }
   return 0;
}
```

Output:
terminate called after throwing an instance of 'char'

This application has requested the Runtime to terminate it in an unusual way. Please contact the application's support team for more information