Output and error tracing question

** Unticked checkboxes are the answers.

1.

```
int [ , ] arr = new int [ , ] { { 1, 2, 3 }, { 4, 5, 6 } }; // line 1

for(int i=0; i<arr.GetLength(0); i++)
{
        for(int j=0; j<arr.GetLength(1); j++)
        {
                Console.Write(arr[i,j]+" ");
        }
}
```

What will be the output of this code?

- ☐ 1 2 3 4 5 6
- ☑ ~~6 5 4 3 2 1~~
- ☑ ~~Error for line 1 because array size (row and column) not given.~~

2.

```
int[ , ] arr = new int [2, 2];
arr[0, 0] = 1; arr[0, 1] = 2; arr[1, 0] = 3; arr[1, 1] = 4;
// data stored in arr like this
// 1 2
// 3 4
for(int i=0; i<arr.GetLength(0); i++)
{
        for(int j=0; j<arr.GetLength(1); j++)
        {
                Console.Write(arr[ j, i ] +" ");
        }
}
```

Output of this code?

- ☑ ~~1 2 3 4~~
- ☑ ~~4 3 2 1~~
- ☐ 1 3 2 4
- ☑ ~~2 4 1 3~~

3. An array in c# can contain elements of multiple data types.

- ☑ ~~True~~
- ☐ False

4. 2d array in c# must have same column length for every row.

- ☐ True
- ☑ ~~False~~

5. Jagged array can contain elements of multiple data types.

- ☑ ~~True~~
- ☐ False

6. How many maximum elements can hold this array,
   int [, , ,] arr = new int [3, 2, 4, 2];

- ☑ ~~(3*2) + (4*2)~~
- ☑ ~~3+2+4+2~~
- ☐ 3*2*4*2
- ☑ ~~(3*2*4)+2~~

7. Which object oriented feature provides reusability?

- ☐ Inheritance
- ☑ ~~Polymorphism~~
- ☑ ~~Abstraction~~
- ☑ ~~Overloading~~

8. Child class can access only the public ,internal and protected property of the parent class.

- ☐ True
- ☑ ~~False~~

9. Parent class can access only the public ,internal and protected property of the child class.

- ☑ ~~True~~
- ☐ False

10.

```
class Parent
{
        public Parent()
        {
                Console.Write("Parent"+" ");
        }
}
class child : Parent
{
        public child()
        {
                Console.Write("Child"+" ");
        }
}
class Program
{

        static void Main(string[] args)
        {
                child c = new child();

                Console.ReadKey();
        }
}
```

Output of the code.

- ☑ ~~Child~~
- ☑ ~~Parent~~
- ☑ ~~Child Parent~~
- ☐ Parent Child

11.

```
class Parent
{
        protected int money;

        public Parent(int m)
        {
                Console.Write("Parent" + " "+money+" ");
                this.money = m;
        }

}
class Child : Parent
{
        public Child(int m):base(m)
        {
                Console.Write("Child"+" "+money+" ");
                this.money = m;
        }
}
class Program
{

        static void Main(string[] args)
        {
                Child c = new Child(500);
                Parent p = new Parent(1000);

                Console.ReadKey();
        }
}
```

What will be the output of this code?

- ☑ ~~Child 500 Parent 1000~~
- ☐ Parent 0 Child 500 Parent 0
- ☑ ~~Parent 500 Child 500 Parent 1000~~
- ☑ ~~Parent 0 Child 0 Parent 0~~

12.

```
class GrandParent
{
        int value;
        public GrandParent(int a)
        {
                this.value = a;
                Console.Write(value + " ");
        }
}
class Parent : GrandParent
{
        int value;
        public Parent(int a):base(a*2)
        {
                this.value = a;
                Console.Write(value + " ");
        }
}
class Child: Parent
{
        int value;
        public Child(int a):base(a*2)
        {
                this.value = a;
                Console.Write(value + " ");
        }
}
class Program
{
        static void Main(string[] args)
        {
                Child c = new Child(2);

                Console.ReadKey();
        }
}
```

Output of the code.

- [x] ~~0 0 0~~
- [x] ~~2 2 2~~
- [x] ~~2 4 8~~
- [ ] 8 4 2