`asp-action` u sljedećem odsječku

```
<a asp-action="Create">Unos nove države</a>
```

je primjer korištenja

a tag-helpera

b css stila

c util-atributa

d vlastite komponente

e parcijalnog pogleda

Iteracija u Scrumu se naziva

a backlog

b shippable

c increment

d game

e sprint

Proces u kojem nakon svake promjene u repozitoriju dolazi do izgradnje programa i pokretanja testova, a zatim i do distrubije programa korisnicima (npr. u smislu objave nove verzije web-aplikacije ili dostupnosti ažuriranja) naziva se

a Continuos publishing

b Continuos delivery

c Commit to publish (CTP)

d Continuos integration

e Push to publish (PTP)

Kako se zove osoba kojoj se pripisuje prva upotreba izraza *programsko inženjerstvo*?

a Charles Babbage

b Edgar Dijkstra

c Ada Lovelace

d Alan Turing

e Margaret Hamilton

Metoda upravljača koja predstavlja neku akciju, a koristi asinkrone metode i await, bit će oblika

a `public async Task<IActionResult> Akcija(...`

b `public Task<IActionResult> Akcija(...`

c `public await Task<IActionResult> Akcija(...`

d `public async IActionResult Akcija(...`

e `public await IActionResult Akcija(...`

Što će se ispisati izvođenjem sljedećeg programa

```
class Razred : IDisposable{
    private string naziv;
    public Razred(string naziv){
        this.naziv = naziv;
    }

    public void Dispose(){
        Console.WriteLine("Dispose: " + naziv);
    }
}

class Program{
    static void Main(string[] args) {
        try {
            Razred r1 = new Razred("A1");
            using (Razred r2 = new Razred("B2")) {
                Razred r3 = new Razred("C3");
                throw new Exception("Poruka");

            }
            r1.Dispose();
        }
        catch { }
    }
}
```

a Dispose: B2 Dispose: A1

b Dispose: A1

c Dispose: A1 Dispose: B2 Dispose: C3

d Dispose: B2

e Dispose: C3 Dispose: B2

Imitiranje izgleda i toka ekrana naziva se

a GUI framing

b wireframing

c GUI mocking

d wireless screen

e wireless framing

Neka su *DayPeriod* i *C* definirani na sljedeći način

```
public enum DayPeriod {
      Morning = 1, Evening = 2, Afternoon = 4, Night = 8
};

class C<T> : where T:struct {
...
```

Označite sve ispravne retke.

a `C<DayPeriod> c = new();`

b `C<string> c = new();`

c `C<List<int>> c = new();`

d `C<List<string>> c = new();`

e `C<int> c = new();`

Koje su vrste zahtjeva?

a Poslovni, sistemski, operativni

b Sistemski, poslovni, rizični, obvezni

c Poslovni, korisnički, funkcionalni, operativni

d Poslovni, korisnički, funkcionalni, nefunkcionalni

e Poslovni, korisnički, sistemski, ne-sistemski

Što od navedenog **nije** naredba Gita

a stash

b commit

c track

d checkout

e merge

Način kreiranja Entity Framework modela u kojem se model dizajnira korištenjem grafičkog sučelja, a baza podataka nastaje na osnovu modela naziva se

a Migration Model

b Model First

c GUI Model

d Forward Model

e Database First

Za prijenos varijabilnog broja argumenata koristi se ključna riječ:

a out

b this

c params

d args

e ref

Označite kriterije kvalitete zahtjeva.

Napomena: Pitanje ima više točnih odgovora

a invarijantnost

b nedvosmislenost

c provjerljivost

d potpunost

e inverznost

Generičko sučelje definirano modifikatorom *out*, npr.

```
public interface IEnumerable<out T>
```

je

a kovarijantno

b kontravarijantno

c invarijantno

d izlazno-varijantno

e varijantno

Neka su definirane sljedeće hijerarhije:

- Vozilo <- Automobil <- ElektricniAutomobil
- Vozilo <- Motocikl
- Osoba <- Djelatnik <- Direktor

i postupak `Djelatnik ZaduzenZa(Automobil a)`

Što je od navedenog ispravno

a `Func<Vozilo, Direktor> func = ZaduzenZa;`

b `Func<Vozilo, Osoba> func = ZaduzenZa;`

c `Func<ElektricniAutomobil, Osoba> func = ZaduzenZa;`

d `Func<ElektricniAutomobil, Direktor> func = ZaduzenZa;`

e `Func<Osoba, ElekricniAutomobil> func = ZaduzenZa;`

Što ispisuje sljedeći programski odsječak?

```
static void Main(string[] args)
{
    Action<Func<int, int, int>> action = (f1) => Console.WriteLine(f1(5, 20));
    action += (f2) => Console.WriteLine(f2(3, 6));
    action((f1, f2) => f1 + f2);
}
```

a
```
5, 20
```

b
```
9
```

c
```
8
26
```

d
```
5, 20
3, 6
```

e
```
25
9
```

Mjerna jedinica za količinu posla je

a `dan`

b `mjesec`

c `čovjek / mjesec`

d `mjesec / čovjek`

e `čovjek * mjesec`

Koja od tvrdnji **nije** istinita:

a Može postojati više finally blokova, tj. po jedan za svaki catch blok.

b Kada dođe do pogreške u try bloku, a postoji više catch blokova, obavlja se prvi catch blok koji obrađuje nastali tip iznimke

c Ako postoji više catch blokova, posljednji se navodi blok koji obrađuje općenite iznimke (tipa System.Exception)

d finally blok se izvodi neovisno o tome da li se dogodila iznimka ili ne.

e Za jedan try blok može postojati jedan ili više catch blokova koji obrađuju različite vrste pogrešaka.

Napomena: Pitanje može imati više točnih odgovora.

```
string connString = ... read from configuration code... ;
string name = ... user input ...
using (var conn = new SqlConnection(connString))  {
  using (var command = conn.CreateCommand()) {
    command.CommandText = "SELECT TOP 1 PersonId FROM Person WHERE LastName LIKE %'" + name + "%'";
    command.Connection = conn;
    conn.Open();
    using (var reader = command.ExecuteReader()) {
        while (reader.Read()) {
          object id = reader[0];
          Console.WriteLine(id);
        }
    }
  }
}
```

a Naredba `command = conn.CreateCommand()` je neispravna i mora se zamijeniti s `command = new SQLCommand()`

b DataReader i/ili veza prema bazi nisu zatvorene u slučaju iznimke.

c Odsječak je primjer koda podložnog *SQL Injectionu*.

d Isti rezultat bi postigli korištenjem naredbe `command.ExecuteScalar` umjesto korištenja *DataReadera*

e Sljedeći redak je neispravan `object id = reader[0];` jer se cjelobrojne vrijednosti (*PersonId* u primjeru) ne mogu dohvatiti uglatim zagradama i pohraniti u varijablu tipa *object*

Povelja projekta je:

a Brošura projekta koja sadrži programsku, tehničku i ostalu prateću dokumentaciju projekta.

b Izjava koju potpisuju svi sudionici projekta.

c Dokument koji opisuje sveukupnu organizaciju projekta.

d Dokument koji sadržava raspored projekta.

e Dokument kojim pokretač projekta ili sponzor odobrava projekt.

Koja tvrdnja **nije** točna?

a Projekt završava u trenutku kada postane jasno da su ciljevi projekta dostignuti ili kada se zaključi da ciljevi projekta ne mogu ili neće biti dostignuti.

b Svaki projekt mora imati jasno određen početak i kraj.

c Projekt se odnosi na rad na nečemu što prije nije postojalo i što se razlikuje od rezultata nastalih sličnim projektima.

d Projekti uglavnom traju više od jedne godine i završavaju kad su svi ciljevi projekta dostignuti.

e Projekti mogu biti kratki ili trajati godinama, ali će svakako završiti.

Ako u konfiguracijskoj datoteci za klijentske biblioteke (npr. bower.json ili packages.json) želimo u budućnosti uključiti noviju verziju jQueryja sve dok je ona oblika 3.minor.patch, a u trenutku pisanja postoji verzija 3.3.1 napisat ćemo

a

```
{
 ...
 "jQuery" : "3.^3.^1",
 ...
}
```

b

```
{
 ...
 "jQuery" : "~3.3.1",
 ...
}
```

c

```
{
 ...
 "jQuery" : "3.*.*",
 ...
}
```

d

```
{
 ...
 "jQuery" : ">3.3.1",
 ...
}
```

e

```
{
 ...
 "jQuery" : "^3.3.1",
 ...
}
```

Verzija konfiguracije koja predstavlja alternativu originalnoj verziji i živi paralelno s njom naziva se

a revizija

b isporuka

c osnovica

d varijanta

e objava

Ako unutar plana projekta kalendar za neku osobu evidentira radno vrijeme od ponedjeljka do petka od 8-12, onda 75% raspoloživosti te osobe znači da je radno vrijeme te osobe

a 4 sata rada tjedno

b 15 sati rada tjedno

c 7,5 sati rada tjedno

d 20 sati rada tjedno

e 30 sati rada tjedno

Ako se unutar pogleda u MVC aplikaciji kao model koristi razred Mjesto koji ima svojstvo Naziv čija se vrijednost ispisuje unutar pogleda, kako izgleda odsječak takvog pogleda?

a

```
@model Mjesto
...
@Mjesto.Naziv
```

b

```
@model Mjesto
...
@Naziv
```

c

```
@model Mjesto
...
@Model.Naziv
```

d

```
@Model Mjesto
...
@Naziv
```

e

```
@Model Mjesto
...
@model.Naziv
```

private protected modifikator pristupa razredima i članovima razreda označava:

a Pristup ograničen na naslijeđene razrede (bez obzira gdje su definirani) i ostale razrede u projektu u kojem je razred definiran

b Pristup ograničen na naslijeđene razrede definirane unutar istog projekta i ostale razrede u projektu u kojem je razred definiran

c Pristup ograničen na razred u kojem je član definiran

d Pristup ograničen na razred i nasljeđene razrede

e Pristup ograničen na program u kojem je razred definiran

Način kreiranja Entity Framework modela u kojem je model opisan kroz ručno napisane razrede (bez grafičkog sučelja), a baza podataka nastaje na osnovu modela naziva se

a Database First

b Model First

c Code First

d Forward Model

e Migration Model

Logička naredba (postupak u C#) kojom se program testira tako da njen uvjet mora biti istinit, a ukoliko nije, program pada, naziva se:

a Točka prekida (Breakpoint)

b Iznimka (Exception)

c Tvrdnja (Assert)

d Događaj (Event)

e Barikada

Koja razina važnosti zapisa se koristi za evidentiranje zapisa trajnijeg karaktera koji služi za praćenje toka rada aplikacije (npr. informacija o posjetu određenoj stranici ili evidencija postavljenih kriterija pretrage)?

a Trace

b Debug

c Warning

d Error

e Information

Konkretizacija pojedinog modela softverskog procesa razradom aktivnosti te uvođenje specifične terminologije i artefakata te propisivanjem stila i organizacija rada opisuje se

a životnim ciklusom razvoja softvera

b metodologijom razvoja softvera

c metodom razvoja softvera

d hijerarhijskom listom zadataka

e planom projekta

Neka su definirane sljedeće hijerarhije:

- Vozilo <- Automobil <- ElektricniAutomobil
- Vozilo <- Motocikl
- Osoba <- Djelatnik <- Direktor

i postupak `void Zaduzi(Automobil a, Djelatnik d)`

Što je od navedenog ispravno?

a `Action<ElektricniAutomobil, Osoba> action = Zaduzi;`

b `Func<ElektricniAutomobil, Osoba> action = Zaduzi;`

c `Action<Vozilo, Direktor> action = Zaduzi;`

d `Action<ElektricniAutomobil, Direktor> action = Zaduzi;`

e `Action<Vozilo, Osoba> action = Zaduzi;`

Želimo li u nekom upitu na bazu podataka korištenjem ADO.NET-a izvršiti UPDATE, INSERT ili DELETE naredbu na nekom objektu tipa DbCommand pozvat ćemo postupak

a SetData

b ExecuteReader

c PostQuery

d ExecuteNonQuery

e ExecuteScalar

Neka su razredi A, B i C definirani na sljedeći način:

```
class A {
    public A(B b, C c)  {
      Console.Write("A ");
    }
  }

  class B {
    public B(C c) {
      Console.Write("B ");
    }
  }

  class C {
    public C()
    {
      Console.Write("C ");
    }
  }
```

a `ServiceProvider` definiran s

```
ServiceProvider BuildDI() {
      IServiceCollection services = new ServiceCollection();
      var provider = services.AddTransient<A>().AddScoped<B>().AddTransient<C>()
                             .BuildServiceProvider();
      return provider;
    }
```

Što će se dogoditi i/ili ispisati pokretanjem sljedećeg programskog odsječka

```
using var serviceProvider = BuildDI();
var scope = serviceProvider.CreateScope();
var x = scope.ServiceProvider.GetRequiredService<A>();
var y = scope.ServiceProvider.GetRequiredService<A>();
```

a C B A

b C B C A C B C A

c C B C A C A

d C B C A

e C B A B A

Kako nazivamo prilagodbu metode razvoja softvera odabirom praksi pojedinih metoda?

a empirical method

b holystic method

c cherry picking

d agile approach

e method tailoring

Neka je zadan razred Vehicle sa sljedećom definicijom:

```
public class Vehicle {
    public string Model { get; set; }
    public double HorsePower { get; set; }
    public string LicencePlate {get; set; }
   ...
```

i neka je zasad sljedeći kod:

```
List<Vehicle> list = ... //load from a source
var x = list.Select(v => new { v.HorsePower})
            .First();
```

Kojeg tipa je varijabla x?

a tipa *Vehicle* pri čemu su *Model* i *LicencePlate null*, a jedina inicijalizirana vrijednost je ona svojstva *HorsePower*

b *double*

c anonimnog razreda sa svojstvom HorsePower

d anonimnog razreda sa svojstvom bez imena

e *double?* zato što je lista mogla biti prazna

Uz pretpostavku da se u tablici Stavka nalazi

- 5 zapisa sa šifrom 1
- 3 zapisa sa šifrom 2
- 2 zapisa sa šifrom 3

sljedeći programski odsječak

```
int sum = 0;
var upit = context.Artikl
                  .Include(a => a.Stavka)
                  .AsNoTracking()
                  .Where(a => a.SifArtikla >=1 && a.SifArtikla <= 3)
                  .ToList();
foreach (Artikl artikl in upit) {
    sum += artikl.Stavka.Count();
}
...
```

generira

a 1 + 5 + 3 + 2 SELECT upita

b 1 + 5 + 1 + 3 + 1 + 2 SELECT upita

c 5 + 3 + 2 SELECT upita

d 1 SELECT upit

e 1 + 3 SELECT upita

U kojoj fazi životnog ciklusa programske potpore se modeliraju podaci?

a planiranje

b analiza

c primjena

d oblikovanje

e implementacija

Ako je u datoteci Startup.cs ASP.NET Core aplikacije u postupku Configure definirana sljedeća ruta

```
app.UseEndpoints(endpoints =>
{
  endpoints.MapDefaultControllerRoute();
});
```

kako izgleda zadnji odsječak url adrese koja bi odgovarala sljedećem postupku iz datoteke MjestoController.cs

```
public IActionResult Edit(int id){...}
```

a .../Home/Index/5

b .../Edit/Mjesto/5

c .../controller=Mjesto/action=Edit/5

d .../Home/Index/Edit/Mjesto/5

e .../Mjesto/Edit/5

U planu projekta vrijedi sljedeća formula

a Posao = Trajanje * Jedinice (Work = Duration * Units)

b Jedinice = Posao * Trajanje (Units = Work * Duration)

c Jedinice = Trajanje + Posao (Units = Duration + Work)

d Posao = Trajanje (Work = Duration)

e Trajanje = Posao * Jedinice (Duration = Work * Units)

Što će se ispisati izvođenjem sljedećeg programskog odsječka

```
int a = 0, b = 0, c;
try {
    try {
        Console.Write("T1 ");
        c = b / a;
    }
    catch (Exception exc) {
        Console.Write("E1 ");
        c = a / b;
    }
    finally {
        Console.Write("F1 ");
    }
}
catch (Exception exc2) {
    Console.Write("E2 ");
}
finally {
    Console.Write("F2 ");
}
```

a T1 E2 F2

b T1 E1 E2 F2

c T1 E1 E2

d T1 E1 F1 E2 F2

e T1 F1 F2