P290 – Развој софтвера

Примери испитних питања за теоријски део писменог дела испита (2021/22)

1. Који од показивача *p1*, *p2* и *p3* није исправно дефинисан у примеру?
```
int* p1,p2;
int* p3=(int*)1000;
```

2. У каквом су односу дужине низова *s1* и *s2* у примеру:
```
char s1[] = "C++";
char s2[] = { 'C', '+', '+' };
```

3. Шта ће исписати овај програм?
```
int a = 1;
int* b=&a;
main(){
      *b = a + 1;
      *b = a + 1;
      cout << a << endl;
}
```

4. Који концепти програмског језика *C++* се употребљавају да би се повећала (потенцијално поновљена) употребљивост написаног кода?

5. Који је редослед уништавања објеката у примеру:
```
int a(3);
main() {
      int* n = new int(10);
      int k(3);
      ...
      delete n;
}
```

6. Да ли је нека (која?) од наредних линија кода неисправна?
```
short* p = new short;
short* p = new short[900];
short* p = new short(900);
short** p = new short[900];
```

7. Које операције класе *Lista* се извршавају у следећој наредби:
```
Lista l2 = Lista();
```

8. Колико пута се позива деструктор класе A у следећем програму?
```
main() {
      A a, b;
      A& c = a;
      A* p = new A();
      A* q = &b;
}
```

9. Шта су шаблони функција?

10. Како се пишу и користе шаблони функција?

11. Написати шаблон функције која рачуна средњу вредност два броја за било који нумерички тип.

12. Шта су шаблони класа?

13. Како се преводе шаблони класа?

14. Шта је неопходан услов да би се позиви неког метода динамички везивали? Шта је довољан услов?

15. Шта је виртуална функција?

16. Шта је чисто виртуална функција?

17. Шта је апстрактна класа?
18. Шта су уметнуте (*inline*) функције? Како се пишу?
19. Шта су уметнути (*inline*) методи? Како се пишу?

20. Која су основна мерила неуспеха при развоју софтвера? Објаснити свако укратко.
21. Које су основне врсте неуспеха при развоју софтвера? Објаснити сваку укратко.
22. Шта је неупотребљив резултат? Који су аспекти неупотребљивости? Објаснити.
23. Који су најчешћи узроци неупотребљивости резултата развоја софтвера? Објаснити укратко.
24. На којим странама се налазе проблеми при развоју софтвера? Објаснити укратко и навести по један пример.
25. Који су најчешћи проблеми на страни заинтересованих лица (улагача) при развоју софтвера? Објаснити укратко.
26. Који су најчешћи проблеми на страни развијаоца при развоју софтвера? Објаснити укратко.
27. Који су најчешћи проблеми који се односе на обе стране у развоју софтвера (улагачи и развојаоци)? Објаснити укратко.
28. Објаснити како приступи планирању могу довести до проблема.
29. Шта је управљање ризицима?
30. Који су основни узроци ризика у развоју софтвера? Навести бар 7.

31. Који су најважнији савремени концепти развоја који су настали из потребе за смањивањем ризика у развојном процесу?
32. Објаснити концепт инкременталног развоја?
33. Објаснити концепт одређивања корака према роковима?
34. Објаснити концепт појачане комуникације међу субјектима?
35. Објаснити концепт давања предности објектима у односу на процесе?
36. Објаснити концепт прављења прототипова?
37. У којим околностима су настале објектно оријентисане развојне методологије?
38. Објаснити основне концепте приступања објектно оријентисаних развојних методологија проблему развоја.
39. Шта је објекат? Објаснити својим речима и навести једну од познатих дефиниција.
40. Шта је класа? Атрибут? Метод?
41. Који су основни концепти на којима почивају технике објектно оријентисаних методологија?
42. Објаснити концепт енкапсулације.
43. Објаснити концепт интерфејса.
44. Објаснити концепт полиморфизма.
45. Објаснити концепт наслеђивања и одговарајуће односе.
46. Кроз које фазе је прошао развој ОО методологија?
47. Које су карактериситике прве фазе развоја ОО методологија?
48. Које су карактериситике друге фазе развоја ОО методологија?

49. Које су карактериситике треће фазе развоја ОО методологија?

50. Шта је УМЛ?
51. Које (три) врсте дијаграма постоје у УМЛ-у? Објаснити.
52. Навести структурне дијаграме УМЛ-а.
53. Која је улога и шта су основни елементи дијаграма класа?
54. Како се представљају односи између класа на дијаграму класа? Који односи постоје?
55. Објаснити кардиналности односа на дијаграму класа – шта представљају и како се означавају?
56. Које су врсте дијаграма класа и по чему се разликују?
57. Која је улога и шта су основни елементи дијаграма компоненти?
58. Која је улога и шта су основни елементи дијаграма објеката?
59. Која је улога и шта су основни елементи дијаграма испоручивања?
60. Која је улога и шта су основни елементи дијаграма пакета?
61. Навести дијаграме понашања УМЛ-а.
62. Која је улога и шта су основни елементи дијаграма активности?
63. Која је улога и шта су основни елементи дијаграма стања?
64. Која је улога и шта су основни елементи дијаграма случајева употребе?
65. Објаснити односе између случајева употребе на дијаграмима случајева употребе.
66. Шта обухвата опис једног случаја употребе?
67. Навести дијаграме интеракција УМЛ-а.
68. Која је улога и шта су основни елементи дијаграма комуникације?
69. Која је улога и шта су основни елементи дијаграма интеракције?
70. Која је улога и шта су основни елементи дијаграма секвенце?

71. Шта је пројекат софтвера?
72. Шта чини пројекат софтвера?
73. Шта је методологија развоја софтвера?
74. Објаснити укратко како изгледа животни циклус пројекта у класичним развојним методологијама?
75. Шта је модел водопада?
76. Какво је место пројектовања у агилним развојним методологијама (укратко)?
77. Шта је дизајн софтвера? Шта је архитектура софтвера? Објаснити сличности и разлике.
78. Шта је апстракција? Објаснити улогу апстракције у дизајну софтвера.
79. Шта је декомпозиција? Објаснити улогу декомпозиције у дизајну софтвера.
80. Које су основне врсте декомпоновања?
81. Шта је функционална декомпозиција?
82. Шта је декомпозиција према променљивости?
83. Шта је логичка декомпозиција?

84. Објаснити однос апстраховања и декомпоновања?
85. Које су пожељне особине софтвера, из угла пројектовања?
86. Шта је флексибилност софтвера?
87. Шта је проширивост софтвера?

88. Шта су кохезија и спрегнутост у контексту развоја софтвера?
89. Навести врсте кохезије у контексту развоја софтвера.
90. Објаснити функционалну кохезију у контексту развоја софтвера.
91. Објаснити секвенцијалну кохезију у контексту развоја софтвера.
92. Објаснити комуникациону кохезију у контексту развоја софтвера.
93. Објаснити процедуралну кохезију у контексту развоја софтвера.
94. Објаснити временску кохезију у контексту развоја софтвера.
95. Објаснити логичку кохезију у контексту развоја софтвера.
96. Објаснити коинцидентну кохезију у контексту развоја софтвера.
97. Навести основне карактеристике спрегнутости у контексту развоја софтвера.
98. Зашто је спрегнутост компоненти софтвера потенцијално проблематична?
99. Навести и укратко објаснити врсте спрегнутости у контексту развоја софтвера.
100. Објаснити спрегу логике у контексту развоја софтвера.
101. Објаснити спрегу типова у контексту развоја софтвера.
102. Објаснити спрегу спецификације у контексту развоја софтвера.
103. Навести и укратко објаснити нивое спрегнутости у контексту развоја софтвера.
104. Објаснити спрегнутост по садржају у контексту развоја софтвера.
105. Објаснити спрегнутост преко заједничких делова у контексту развоја софтвера.
106. Објаснити спољашњу спрегнутост у контексту развоја софтвера.
107. Објаснити спрегнутост преко контроле у контексту развоја софтвера.
108. Објаснити спрегнутост преко маркера у контексту развоја софтвера.
109. Објаснити спрегнутост преко података у контексту развоја софтвера.
110. Објаснити појам *ширина спреге* у контексту развоја софтвера.
111. Објаснити појам *смер спреге* у контексту развоја софтвера.
112. Објаснити појам *статичка спрегнутост* у контексту развоја софтвера.
113. Објаснити појам *динамичка спрегнутост* у контексту развоја софтвера.
114. Објаснити однос статичке и динамичке спрегнутости у контексту развоја софтвера.
115. Објаснити појам *интензитет спрегнутости* у контексту развоја софтвера.
116. На који начин се може приступити мерењу и рачунању интензитета спрегнутости?
117. Навести и објаснити два основна правила у вези спрегнутости компоненти у контексту развоја софтвера.
118. Навести неколико уобичајених начина спрегнутости компоненти.

119. Објаснити карактеристике спрегнутости у случају архитектуре клијент-сервер.
120. Објаснити карактеристике спрегнутости у случају хијерархије припадности.
121. Објаснити карактеристике спрегнутости у случају циркуларне спрегнутости.
122. Објаснити карактеристике спрегнутости у случају спреге путем интерфејса.
123. Објаснити карактеристике спрегнутости у случају спреге путем параметара метода.
124. Објаснити однос концепта класе и појма кохезије у контексту ОО развоја софтвера.
125. Објаснити однос концепта класе и појма спрегнутости у контексту ОО развоја софтвера.

126. Шта је образац за пројектовање? Чему служи?
127. Који су основни елементи образаца за пројектовање? Објаснити их.
128. Објаснити *име*, као елемент обрасца за пројектовање?
129. Објаснити *проблем*, као елемент обрасца за пројектовање?
130. Објаснити *решење*, као елемент обрасца за пројектовање?
131. Објаснити *последице*, као елемент обрасца за пројектовање?
132. Навести шта све обухвата опис једног обрасца за пројектовање?
133. Како су класификовани обрасци за пројектовање? Навести по један пример од сваке врсте образаца.
134. Објаснити намену градивних образаца за пројектовање.
135. Навести бар четири градивна образаца за пројектовање.
136. Објаснити намену структурних образаца за пројектовање.
137. Навести бар пет структурних образаца за пројектовање.
138. Објаснити намену образаца понашања.
139. Навести бар седам образаца понашања.
140. Објаснити када се и како примењује образац Производни метод (*Factory Method*).
141. Скицирати дијаграм класа обрасца за пројектовање Производни метод (*Factory Method*).
142. Објаснити када се и како примењује образац Стратегија.
143. Скицирати дијаграм класа обрасца за пројектовање Стратегија.
144. Објаснити када се и како примењује образац Декоратер.
145. Скицирати дијаграм класа обрасца за пројектовање Декоратер.
146. Објаснити када се и како примењује образац Сложени објекат (Састав, *Composite*).
147. Скицирати дијаграм класа обрасца за пројектовање Сложени објекат (Састав, *Composite*).
148. Објаснити када се и како примењује образац Уникат (*Singleton*).
149. Скицирати дијаграм класа обрасца за пројектовање Уникат (*Singleton*).
150. Објаснити када се и како примењује образац Посетилац (*Visitor*).
151. Скицирати дијаграм класа обрасца за пројектовање Посетилац (*Visitor*).
152. Објаснити када се и како примењује образац Посматрач (*Observer*).
153. Скицирати дијаграм класа обрасца за пројектовање Посматрач (*Observer*).
154. Објаснити када се и како примењује образац Апстрактна фабрика (*Abstract Factory*).

155. Скицирати дијаграм класа обрасца за пројектовање Апстрактна фабрика (*Abstract Factory*).

156. Навести скупове принципа дизајнирања софтвера које смо обрадили?
157. Навести кључне принципе ОО дизајна.
158. Објаснити принцип јединствене одговорности.
159. Објаснити принцип отворености и затворености.
160. Објаснити принцип заменљивости.
161. Објаснити принцип раздвајања интерфејса.
162. Објаснити принцип инверзне зависности.
163. Навести принципе додељивања одговорности (дизајн софтвера).
164. Обаснити принцип дизајна софтвера Информациони експерт.
165. Обаснити принцип дизајна софтвера Стваралац.
166. Обаснити принцип дизајна софтвера Висока кохезија.
167. Обаснити принцип дизајна софтвера Ниска спрегнутост.
168. Обаснити принцип дизајна софтвера Контролер.
169. Обаснити принцип дизајна софтвера Полиморфизам.
170. Обаснити принцип дизајна софтвера Измишљотина.
171. Обаснити принцип дизајна софтвера Индирекција.
172. Обаснити принцип дизајна софтвера Изоловане променљивости.
173. Навести принципе обликовања целина (дизајн софтвера).
174. Објаснити принцип еквивалентности издања и употребе.
175. Објаснити принцип заједничке затворености.
176. Објаснити принцип заједничке употребе.
177. Објаснити принцип стабилне зависности.
178. Објаснити принцип стабилне апстракције.
179. Објаснити принцип ацикличних зависности.

180. Шта је Агилни развој софтвера?
181. Навести основне претпоставке Манифеста агилног развоја.
182. Објаснити претпоставку агилног развоја да су појединци и сарадња испред процеса и алата.
183. Објаснити претпоставку агилног развоја да је функционалан софтвер испред исцрпне документације.
184. Објаснити претпоставку агилног развоја да је сарадња са клијентом испред преговарања
185. Објаснити претпоставку агилног развоја да је реаговање на промене испред праћење плана.
186. Навести бар 8 принципа агилног развоја софтвера.
187. Навести бар 3 методологије агилног развоја софтвера.
188. Шта је екстремно програмирање?

189. Навести бар 8 метода екстремног програмирања.

190. Објаснити метод екстремног програмирања „Клијент је члан тима“.

191. Објаснити метод екстремног програмирања „Корисничке целине (*user stories*)“.

192. Објаснити метод екстремног програмирања „Кратки циклуси“.

193. Објаснити метод екстремног програмирања „Тестови прихватљивости“.

194. Објаснити метод екстремног програмирања „Програмирање у пару“.

195. Објаснити метод екстремног програмирања „Развој вођен тестовима“.

196. Објаснити метод екстремног програмирања „Колективно власништво“.

197. Објаснити метод екстремног програмирања „Непрекидна интеграција“.

198. Објаснити метод екстремног програмирања „Уздржан ритам“.

199. Објаснити метод екстремног програмирања „Отворен радни простор“.

200. Објаснити метод екстремног програмирања „*Игра* планирања“.

201. Објаснити метод екстремног програмирања „Једноставан дизајн“.

202. Објаснити метод екстремног програмирања „Рефакторисање“.

203. Објаснити метод екстремног програмирања „Метафора“.

204. Шта је „Развој вођен тестовима“?

205. Навести и објаснити врсте тестова софтвера.

206. Навести и објаснити укратко основне принципе развоја вођеног тестовима.

207. Објаснити принцип развоја вођеног тестовима „Тестови претходе коду“ и начин његове примене.

208. Објаснити принцип развоја вођеног тестовима „Систематичност“.

209. Навести основне улоге тестова.

210. Објаснити улогу тестова као вида верификације софтвера.

211. Објаснити улогу тестова у оквиру рефакторисања.

212. Објаснити улогу тестова у контексту угла посматрања кода.

213. Објаснити улогу тестова као вида документације.

214. Шта може бити јединица кода која се тестира?

215. Шта може бити предмет тестирања јединице кода?

216. Навести бар 5 библиотека за тестирање јединица кода у програмском језику *C++*.

217. Описати укратко основне могућности библиотеке *CppUnit*.

218. Који су основни елементи које програмер прави при прављењу тестова уз примену библиотеке *CppUnit*? Како?

219. Шта је *Test suit*?

220. Навести основне врсте претпоставки које подржава библиотека *CppUnit*?

221. Описати укратко основне могућности библиотеке *Catch*. Написати пример теста.

222. Који су основни елементи које програмер прави при прављењу тестова уз примену библиотеке *Catch*? Како? Написати пример теста.

223. Шта је *Test case*? Шта је *Test case section*? (*Catch*)

224. Навести основне врсте претпоставки које подржава библиотека *Catch*?

225. Шта су тестови прихватљивости?

226. По чему се тестови прихватљивости разликују од тестова јединица кода?

227. Ко од учесника у развоју софтвера пише тестове јединица кода? А тестове прихватљивости?

228. Навести основне врсте полиморфизма.

229. Шта је хијерархијски полиморфизам?

230. Шта је параметарски полиморфизам?

231. Шта је имплицитни полиморфизам?

232. Шта је ад-хок полиморфизам?

233. Шта су шаблони функција?

234. Шта су шаблони класа?

235. Шта су функционални објекти – функционали?

236. Какав је однос рефакторисања и писања програмског кода?

237. Шта су основни мотиви за рефакторисање кода?

238. Када се приступа рефакторисању кода?

239. На основу чега се одлучује да је потребно рефакторисати неки код?

240. Набројати бар 10 слабости кода (тзв. *заударања*) које указују да би требало размотрити рефакторисање?

241. Зашто је добро елиминисати понављања из кода? (рефакторисање)

242. Зашто дугачки методи могу представљати проблем? (рефакторисање)

243. Зашто велика класа може да представља проблем? (рефакторисање)

244. Шта су *дивергентне промене*? Зашто су проблематичне? (рефакторисање)

245. Шта је *дистрибуирана апстракција*? Зашто је проблематична? (рефакторисање)

246. Зашто велика зависност неке класе или метода од других класа може да представља проблем? (рефакторисање)

247. Зашто наредба *switch* може да представља проблем? (рефакторисање)

248. Шта је *спекулативно уопштавање*? Зашто може да представља проблем? (рефакторисање)

249. Зашто привремене променљиве могу да представљају проблем? (рефакторисање)

250. Зашто ланци порука могу да представљају проблем? (рефакторисање)

251. Зашто постојање класе посредника може да представља проблем? (рефакторисање)

252. Шта је *непожељна блискост*? Зашто је проблематична? (рефакторисање)

253. Какав је однос агилног развоја софтвера и писања коментара? Зашто коментари могу да буду мотив за рефакторисање?

254. Шта би требало да садржи опис сваког од рефакторисања у каталогу?

255. Навести бар 5 група техника рефакторисања.

256. Навести бар 5 техника рефакторисања.

257. У којим случајевима рефакторисање може бити значајно отежано?
258. Како и зашто може бити отежано рефакторисање у присуству базе података?
259. Зашто може бити отежано рефакторисање спољног интерфејса неке класе?
260. У којим случајевима рефакторисање може да не представља добро решење?
261. Какав је однос рефакторисања и перформанси софтвера?

262. Шта су багови?
263. Навести једну класификацију багова и објаснити је.
264. Шта су *неконзистентности у корисничком интерфејсу* и какве узроке и последице имају?
265. Шта су *неиспуњена очекивања* и какве узроке и последице имају?
266. Објаснити проблем слабих перформанси и могуће узроке.
267. Које околности посебно погодују настанку багова? Објаснити две.
268. Које околности смањују вероватноћу настајања багова? Објаснити две.
269. Које околности олакшавају проналажење узрока багова? Објаснити две.
270. Који су основни приступи проблему дебаговања? Објаснити укратко њихов однос.
271. Описати емпиријски (научни) метод дебаговања.
272. Објаснити хеуристичко дебаговање.
273. Навести бар 6 основних правила за дебаговање (по Д.Ејгенсу).
274. Објаснити правило дебаговања „Разумети систем“.
275. Објаснити правило дебаговања „Навести систем на грешку“.
276. Објаснити правило дебаговања „Најпре посматрати па тек затим размишљати“.
277. Објаснити правило дебаговања „Подели па владај“.
278. Објаснити правило дебаговања „Правити само једну по једну измену“.
279. Објаснити правило дебаговања „Правити и чувати трагове извршавања“.
280. Објаснити правило дебаговања „Проверавати и наизглед тривијалне ствари“.
281. Објаснити правило дебаговања „Затражити туђе мипљење“.
282. Објаснити правило дебаговања „Ако нисмо поправили баг, онда он није поправљен“.
283. Навести најважније технике за превенцију настајања багова.
284. Навести основне унутрашње технике и алате за дебаговање.
285. Објаснити *писање претпоставки* као технику превенцију настајања багова.
286. Објаснити технику *остављања трагова при извршавању* као превенцију настајања багова.
287. Објаснити *коментарисање кода* као технику превенцију настајања багова.
288. Објаснити *тестирања јединица кода* као технику превенцију настајања багова.
289. Нвести најважније спољашње технике и алате за дебаговање.
290. Навести основне технике употребе дебагера.
291. Објаснити технику употребе дебагера „Извршавање корак по корак“.
292. Објаснити технику употребе дебагера „Постављање тачака прекида“.

293. Објаснити технику употребе дебагера „Праћење вредности промењљивих“.

294. Објаснити технику употребе дебагера „Праћење локалних промењљивих“.

295. Објаснити технику употребе дебагера „Праћење стања стека“.

296. Објаснити технику употребе дебагера „Праћење рада на нивоу инструкција и стања процесора“.

297. Шта је *конкурентно извршавање*?

298. Објаснити појам *паралелно извршавање*.

299. Објаснити појам *дистрибуирано извршавање*.

300. Објаснити појам *процес*.

301. Објаснити појам *нит*.

302. Зашто се уводи концепт нити, ако већ постоји концепт процеса?

303. Објаснити сличности и разлике нити и процеса.

304. Објаснити како се програмирају нити помоћу стандардне библиотеке *C++*-а. Класе, методи,…

305. Објаснити како се програмирају нити помоћу библиотеке *QT*. Класе, методи,...

306. Навести и укратко објаснити основне операције са нитима.

307. Објаснити детаљно операцију прављења нити. Како се имплементира помоћу стандардне библиотеке *C++*-а?

308. Објаснити детаљно операцију прављења нити. Како се имплементира помоћу библиотеке *QT*?

309. Објаснити детаљно операцију довршавања нити. Како се имплементира помоћу стандардне библиотеке *C++*-а?

310. Објаснити детаљно операцију довршавања нити. Како се имплементира помоћу библиотеке *QT*?

311. Објаснити детаљно операције суспендовања и настављања нити. Како се имплементирају стандардне библиотеке *C++*-а?

312. Објаснити детаљно операције суспендовања и настављања нити. Како се имплементирају помоћу библиотеке *QT*?

313. Објаснити детаљно операцију прекидања нити. Како се имплементира помоћу стандардне библиотеке *C++*-а?

314. Објаснити детаљно операцију прекидања нити. Како се имплементира помоћу библиотеке *QT*?

315. Објаснити детаљно операцију чекања нити. Како се имплементира помоћу стандардне библиотеке *C++*-а?

316. Објаснити детаљно операцију чекања нити. Како се имплементира помоћу библиотеке *QT*?

317. Који су најважнији проблеми при писању конкурентних програма?

318. Шта је *мутекс*? Како се употребљава?

319. Објаснити подршку за мутексе у оквиру стандардне библиотеке *C++*-а?

320. Објаснити подршку за мутексе у оквиру библиотеке *QT*.

321. Шта је, чему служи и како се користи *lock_guard* из стандардне библиотеке *C++*-а? Објаснити детаљно.

322. Чему служи класа *QMutexLocker* библиотеке *QT*? Објаснити детаљно.

323. Шта су *катанци*? Како се употребљавају?

324. Објаснити подршку за катанце у оквиру стандардне библиотеке *C++*-а?

325. Објаснити подршку за катанце у оквиру библиотеке *QT*.

326. Чему служи класа *QReadLocker* библиотеке *QT*? Објаснити детаљно.

327. Чему служи класа *QWriteLocker* библиотеке *QT*? Објаснити детаљно.

328. Шта је синхронизација? Шта се све може синхронизовати у конкурентним програмима?

329. Шта су *семафори*?

330. Објаснити подршку за семафоре у оквиру стандардне библиотеке *C++*-а.

331. Објаснити подршку за семафоре у оквиру библиотеке *QT*.

332. Објаснити функцију *async* и шаблон *future* стандардне библиотеке *C++*-а.

333. Објаснити шаблоне *future* и *promise* стандардне библиотеке *C++*-а.

334. Какви могу бити потпрограми у контексту конкурентног програмирања?

335. Шта су потпрограми са јединственим позивањем?

336. Шта су потпрограми са поновљивим позивањем?

337. Шта су потпрограми безбедни по нити?

338. Какве могу бити класе у контексту конкурентног програмирања? Објаснити.

339. Које су најчешће грешке при писању конкурентних програма? Објаснити.

340. Навести неке начине развоја програма који омогућавају безбедно писање конкурентних програма? Објаснити.

341. Када долази на ред старање о понашању кода у конкурентном окружењу?

342. Како се бира где се и како постављају мутекси и катанци?

343. Навести основне видове међупроцесне комуникације.

344. Објаснити разлику између комуникације међу процесима и комуникације међу нитима.

345. Шта су архитектуре засноване на догађајима?

346. Објаснити мотивацију за употребу архитектура заснованих на догађајима.

347. Објаснити основне појмове и концепте архитектуре засноване на догађајима.

348. Навести и објаснити слојеве тока догађаја код архитектура заснованих на догађајима.

349. Објаснити основне концепте примене архитектуре засноване на догађајима у оквиру библиотеке *QT*.

350. Објаснити концепт сигнала у контексту примене примене архитектуре засноване на догађајима у оквиру библиотеке *QT*.

351. Објаснити концепт слотова у контексту примене примене архитектуре засноване на догађајима у оквиру библиотеке *QT*.

352. Објаснити повезивање сигнала и слотова у случају примене примене архитектуре засноване на догађајима у оквиру библиотеке *QT*.

353. Објаснити однос архитектура заснованих на догађајима и проблема кохезије и спрегнутости.

354. Шта је *софтверска метрика*?

355. Навести најважније типове софтверских метрика?

356. Објаснити врсте метрика у развоју софтвера.

357. Навести неколико метрика праћења развоја софтвера. Шта оне описују?

358. Објаснити метрику напретка и како се обично користи.

359. Нацртати пример дијаграма праћења напретка са 4 криве.

360. Навести неколико метрика дизајна софтвера. Шта оне описују?

361. Објаснити метрику *стабилност пакета*.

362. Објаснити метрику *апстрактност пакета*.

363. Објаснити однос метрика стабилности и апстрактности пакета.

364. Објаснити метрику *функционална кохезија пакета*?

365. Шта је цикломатичка сложеност?

366. Шта је Холстедова сложеност?

367. Шта су системи за контролу верзија? Објаснити.

368. Објаснити архитектуру и навести основне операције при раду са системима за контролу верзија.

369. Шта је спремиште? Шта садржи? Како је организовано?

370. Шта је радна копија, у контексту употребе система за контролу верзија?

371. Објаснити појам ознаке у контексту употребе система за контролу верзија.

372. Објаснити гранање у контексту употребе система за контролу верзија.

373. Шта су конфликти и како се решавају, у контексту употребе система за контролу верзија?

374. Шта је спајање верзија, у контексту употребе система за контролу верзија?

375. Објаснити пример стратегије означавања верзија.

376. Шта су системи за праћење задатака и багова? Објаснити намену и основне елементе.

377. Навести и укратко објаснити основне концепте система *Redmine*.

378. Објаснити улогу стања картица и начин њиховог мењања (на примеру система *Redmine*).

379. Шта чини документацију софтвера?

380. Коме је и зашто потребна документација?

381. Навести и укратко објаснити основне оправдане и неоправдане мотиве за прављење документације.

382. Објаснити улогу документације као вида спецификације захтева пројекта.

383. Објаснити улогу документације као средства за комуникацију.

384. Објаснити улогу документације у разматрању недоумица у пројекту.

385. Објаснити поделу документације по намени.

386. Шта обухвата корисничка документација софтвера?

387. Шта обухвата техничка (системска) документација софтвера?

388. Какав је однос агилног развоја софтвера према писању документације? Који видови документације се подстичу а који не?

389. Шта су алати за *унутрашње* документовање програмског кода? Зашто су потребни и по чему се суштински разликују од одржавања спољашње документације?

390. Шта је *Doxygen*? Шта омогућава? Навести примере анотације кода.

391. Шта је оптимизација софтвера?

392. Које су информације неопходне за успешну оптимизацију?

393. Објаснити „оптимизацију унапред“. Добре и лоше стране?

394. Објаснити „оптимизацију уназад“. Добре и лоше стране?

395. Каква је суштинска разлика између оптимизација унапред и уназад? Када је боље применити коју од њих?

396. Који основни проблем производи примена оптимизације у агилном развоју софтвера? Како се превазилази?

397. Како се деле технике оптимизације? Навести неколико примера.

398. Навести бар 7 општих техника оптимизације кода.

399. Објаснити технике оптимизације „одбацивање непотребне прецизности“ и „таблице унапред израчунатих вредности“.

400. Објаснити технике оптимизације „интеграција петљи“, „измештање инваријанти изван петље“ и „размотавање петљи“.

401. Објаснити технике оптимизације „смањити број аргумената функције“ и „избегавати глобалне промењљиве“.

402. Објаснити технике оптимизације „употреба уметнутих функција“ и „елиминација гранања и петљи“.

403. Објаснити технике оптимизације „замењивање динамичког услова статичким“ и „снижавање сложености операције“.

404. Објаснити технике оптимизације „редослед проверавања услова“ и „избор решења према најчешћем случају“.

405. Које су најчешће грешке при оптимизацији? Објаснити.

406. Навести и укратко објаснити три технике оптимизације специфичне за програмски језик C++.

407. Шта су „оптимизације у ходу“? Навести примере.

408. Шта су профајлери? Чему служе? Шта пружају програмерима?