Конкурентно и дистрибуирано програмирање *11. фебруар 2017.*

1. (20) К Код *Tie breaker* алгоритма за n процеса се догодио следећи случај – приликом извршавања кода за улазак у критичну секцију, свих n процеса су ушли у стање 1 и ниједан још није ушао у стање 2. Одговорите на следећа питања: а) Да ли процес који је први ушао у стање 1 први улази у стање 2? б) Да ли процес који је први ушао у стање n-2 први улази у критичну секцију? в) Да ли процес који је последњи ушао у стање 1 може да буде трећи који улази у стање 2? г) Да ли процес који је последњи ушао у стање 1 може да буде први који улази у критичну секцију?

2. (20) К У берберници раде два берберина, Аца и Браца, постоји 10 столица за чекање и још петоро муштерија може да стоји и чека. Муштерије које долазе се изјашњавају да ли чекају код Аце или Браце или им је свеједно ко ће да их услужи. Ако муштерија види да нема места у берберници и не може бити услужена, одлази. Када је берберин слободан, муштерија која је најдуже чекала ће прва бити услужена (осим ако чека на другог берберина и тада тражимо следећу муштерију у низу). Када се ослободи столица за чекање, муштерија која је најдуже стајала треба да седне. Уколико је неки од берберина беспослен, он спава, и прва муштерија која код њега дође на ред треба да га пробуди и буде услужена. Користећи мониторе са *signal and wait* дисциплином решити овај проблем.

3. (20) Филтерски процеси имају један улаз и један излаз и раде следеће: примају позитивне вредности на улазу и прослеђују их на излаз ако су веће од запамћеног минимума процеса. Процеси имају само две локације, за сачувани минимум и за задњу примљену вредност. Када на улаз стигне EOS, избацују минималну вредност на излаз и затим EOS. Направите *pipeline* од n процеса који опадајуће сортирају: до n улазних позитивних вредности које се убацују на почетак *pipeline*-а, а завршавају се са EOS.

4. (20) На *Ethernet* локалној мрежи налази се N рачунара од којих сваки има свој диск. Потребно је обезбедити да сваки рачунар може да затражи неки диск простор, тако што би добио расположиве делове диск простора од других рачунара на локалној мрежи. Након добијања информације од осталих рачунара о расположивом простору потребно је послати информацију о резервацији свим рачунарима од којих се узима простор. На крају они јављају потврду о резервацији. Користећи *C-Linda* напишите костур таквог програма тако да важи FIFO принцип резервација. Узети у обзир и случајеве када нема довољно простора на диску на осталим рачунарима.

*Испит траје **3** сата.*

Конкурентно и дистрибуирано програмирање *11. фебруар 2017.*

1. (20) К Код *Tie breaker* алгоритма за n процеса се догодио следећи случај – приликом извршавања кода за улазак у критичну секцију, свих n процеса су ушли у стање 1 и ниједан још није ушао у стање 2. Одговорите на следећа питања: а) Да ли процес који је први ушао у стање 1 први улази у стање 2? б) Да ли процес који је први ушао у стање n-2 први улази у критичну секцију? в) Да ли процес који је последњи ушао у стање 1 може да буде трећи који улази у стање 2? г) Да ли процес који је последњи ушао у стање 1 може да буде први који улази у критичну секцију?

2. (20) К У берберници раде два берберина, Аца и Браца, постоји 10 столица за чекање и још петоро муштерија може да стоји и чека. Муштерије које долазе се изјашњавају да ли чекају код Аце или Браце или им је свеједно ко ће да их услужи. Ако муштерија види да нема места у берберници и не може бити услужена, одлази. Када је берберин слободан, муштерија која је најдуже чекала ће прва бити услужена (осим ако чека на другог берберина и тада тражимо следећу муштерију у низу). Када се ослободи столица за чекање, муштерија која је најдуже стајала треба да седне. Уколико је неки од берберина беспослен, он спава, и прва муштерија која код њега дође на ред треба да га пробуди и буде услужена. Користећи мониторе са *signal and wait* дисциплином решити овај проблем.

3. (20) Филтерски процеси имају један улаз и један излаз и раде следеће: примају позитивне вредности на улазу и прослеђују их на излаз ако су веће од запамћеног минимума процеса. Процеси имају само две локације, за сачувани минимум и за задњу примљену вредност. Када на улаз стигне EOS, избацују минималну вредност на излаз и затим EOS. Направите *pipeline* од n процеса који опадајуће сортирају: до n улазних позитивних вредности које се убацују на почетак *pipeline*-а, а завршавају се са EOS.

4. (20) На *Ethernet* локалној мрежи налази се N рачунара од којих сваки има свој диск. Потребно је обезбедити да сваки рачунар може да затражи неки диск простор, тако што би добио расположиве делове диск простора од других рачунара на локалној мрежи. Након добијања информације од осталих рачунара о расположивом простору потребно је послати информацију о резервацији свим рачунарима од којих се узима простор. На крају они јављају потврду о резервацији. Користећи *C-Linda* напишите костур таквог програма тако да важи FIFO принцип резервација. Узети у обзир и случајеве када нема довољно простора на диску на осталим рачунарима.

*Испит траје **3** сата.*