جلسه اول

شبیه سازی کامپیوتری سیستم ها

کتاب شبیه سازی کامپیوتری سیستم ها حسن صالحی

سر فصل

۱- تعاریف 3+3 نمره

۲- شبیه سازی به روش زمانبندی پیش آمده به زبان برنامه نویسی عمومی c++ 7 نمره

۳- شبیه سازی به روش پردازش فرآیندها به زبانهای مخصوص شبیه سازی gpss 7 نمره

میان ترم 3 نمره تعریف + 7 نمره شبیه سازی به زبان c++ (حذفی)

پایان ترم 3 نمره تعریف + 7 نمره شبیه سازی gpss

2 نمره حضور در کلاس 2 نمره تمرین و پروژه

جلسه دوم

تعریف سیستم:

عبارت است از مجموعه ای از اشیاء یا موجودیت ها و روابطی که بین اشیاء برقرار است. و قواعدی قوانینی که حاکم بر اشیاء و روابط بین آنها.

اجزای سیستم:

یک سیستم ممکن است یک محیط برای سیستم های داخلی کوچکتری بشمار آید که به این سیستم ها محدوده یا اجزای سیستم گفته می شود.

تعریف محیط سیستم:

اغلب تغییرات در یک سیستم بر اثر عوامل خارج سیستم صورت می گیرد به این مجموعه از عوامل که بر

Year. Month. Date. ( )

روی یک سیستم تأثیر می‌گذارند و یا از آن تأثیر می‌پذیرند محیط سیستم گفته می‌شود.

تعریف وضعیت سیستم:

به چگونگی اشیاء و مشخصه‌های اشیاء و روابط بین اشیاء یک سیستم در یک لحظه زمانی وضعیت سیستم در آن لحظه گفته می‌شود.

بدیهی است در یک مطالعه ریاضی وضعیت سیستم بوسیله مقادیر عددی که به اشیاء و مشخصه‌های اشیاء نسبت داده می‌شود مشخص می‌گردد.

در جدول زیر نمونه‌هایی از سیستم‌ها، اشیاء سیستم‌ها و مشخصه‌های اشیاء سیستم‌ها آورده شده است:

سیستم	اشیاء	مشخصه‌های اشیاء
بانک	باجه	تعداد - نوع - فضا
	کارمندان	تعداد - سرعت - مهارت - تخصص - سمت
	مشتری	تعداد - فاصله زمانی بین ورود - موجودی - پارتی - نوع کار
	دستگاه‌ها	تعداد - نوع - میزان خرابی - سرعت
بیمارستان	بیمار	تعداد - نوع بیماری - مدت بستری - میزان رابطه
	پزشک	تعداد - تخصص - شیفت - نرخ عمل
	کارکنان	تعداد - تخصص - شیفت کاری - مهارت
	دستگاه‌ها	تعداد - نوع - میزان خرابی - وضعیت
	تخت‌ها	تعداد - تعداد اتاق - تعداد تخت - وسعت
لیگ فوتبال	تیم	تعداد - رنگ لباس - بودجه
	مربی	تعداد - درجه - سابقه
	بازیکن	تعداد تیم - سن - مبلغ قرارداد
	داور	
	استادیوم	

تعریف متغیرهای سیستم:

اگر در مثال های فوق به مشخصات اشیاء توجه کنیم درمی یابیم که بعضی از مشخصات اشیاء در طول عملکرد سیستم تغییر می یابد و مقادیر مختلفی به خود می گیرد. به همین لحاظ به مشخصات اشیاء یک سیستم متغیرهای داخلی یک سیستم نیز گفته می شود. و به مقدار این متغیرها در یک لحظه معین وضعیت سیستم در آن لحظه گفته می شود.

تعریف متغیرهای تصمیم:

بعضی از متغیرهای سیستم توسط تصمیم گیران یا مدیران سیستم قابل کنترل می باشند. به این متغیرها متغیرهای تصمیم گفته می شود.

برای مطالعه و تجزیه و تحلیل یک سیستم نیاز به مطالعه تغییرات در طول زمان می باشد. که با دو عامل اندازه گیری می شود:

1- حجم یا بزرگی تغییر که برابر است با تفاضل مقادیر یک مشخصه از سیستم در یک فاصله زمانی.

2- درنگ یا تأخیر که برابر است با فاصله زمانی سپری شده از لحظه دریافت محرک توسط سیستم تا زمان بروز تغییر در سیستم.

دو عامل فوق نحوه عکس العمل سیستم با نسبت به یک محرک نشان می دهند به همین لحاظ به آنها پاسخ یا عکس العمل سیستم گفته می شود.

انواع سیستم:

سیستم ها را می توان بصورت های مختلف دسته بندی نمود. از نظر نحوه تغییر وضعیت سیستم نسبت به زمان سیستم ها به دو دسته تقسیم می شوند:

1- سیستم های گسسته 2- سیستم های پیوسته

اگر تغییرات مشخصه های اشیاء سیستم نسبت به زمان گسسته باشد یا در لحظات متمایزی از زمان این تغییرات صورت گیرند سیستم را گسسته می نامند. بطور مثال در یک سیستم بانک تعداد

Year. Month. Date. ( )

مشتریان یکی از مشخصه های اشیاء می باشد که در لحظات متمایزی از زمان با ورود یا خروج یک مشتری تغییر می یابد.

اگر تغییر یکی از مشخصه های اشیاء سیستم نسبت به زمان بصورت متصل یا پیوسته انجام گیرد سیستم را پیوسته گویند. بطور مثال در یک سیستم پالایشگاه حجم مایع پالایش شده یکی از مشخصه های اشیاء است که بصورت پیوسته تغییر می یابد.

مدل سازی یا Modeling :

برای مطالعه و تجزیه و تحلیل سیستم ها روشهای مختلفی وجود دارد نظیر:

1- تجزیه و تحلیل ریاضی

2- بررسی آزمایشگاهی

3- مشاهده عینی و تجربی

4- شبیه سازی فیزیکی

در مطالعه عینی و تجربی یک سیستم آزمایش ها و تغییر متغیرهای سیستم بطور مستقیم بر روی سیستم واقعی صورت می گیرد. ولی از این روش نمی توان برای مطالعه همه سیستم ها استفاده کرد زیرا

1- سیستم هایی که تغییر یک متغیر در یکی از اشیاء آنها باعث دگرگونی کل سیستم می گردد نمی توان از این روش استفاده کرد.

2- ایجاد تغییر برای مشاهده عکس العمل سیستم در برخی از سیستم ها امکان پذیر نمی باشد.

3- در سیستم هایی که در مرحله طراحی می باشند و هنوز وجود خارجی ندارند نمی توان از این روش برای مطالعه و تجزیه و تحلیل استفاده نمود.

در چنین مواقعی یک مدل از سیستم که ترکیبی مناسبی از اشیاء و خصوصیات اشیاء آن سیستم می باشد را تعریف می کنیم و برای بررسی و مطالعه سیستم مورد استفاده قرار می دهیم.
معمولاً نوع بررسی تعیین کننده جزئیات و خصوصیات و میزان اطلاعات موجود در مدل می باشد. بنابراین ممکن است در بررسی های مختلف یک سیستم مدل های مختلفی مورد استفاده قرار گیرد.

جلسه سوم

ادامه صحبت Modeling

هرچه جزئیات بیشتری از یک مدل در سیستم گنجانده شود مدل شباهت بیشتری به سیستم واقعی خواهد داشت. رفتار آن را بهتر نمایش خواهد داد و نتیجه ای که از مطالعه و بررسی مدل به دست می آید به واقعیت نزدیک تر خواهد بود. از طرف دیگر هرچه جزئیات بیشتری در مدل در نظر گرفته شود باعث پیچیده تر شدن مدل سازی و مطالعه و کسب نتایج از آن می گردد.

انواع مدل ها:
مدل ها را می توان به صورت های مختلف دسته بندی کرد. از یک دیدگاه مدل ها را می توان به دو نوع ریاضی و فیزیکی تقسیم بندی نمود. از یک دیدگاه دیگر به دو نوع ایستا و پویا و از یک دیدگاه دیگر به قطعی و احتمالی تقسیم بندی می توان نمود.

مدل های ریاضی خود به دو دسته عددی و تحلیلی یا دو نوع سمبلیک یا گرافیکی دسته بندی می شوند.

مدل های ایستا وضعیت سیستم یا رفتار سیستم را در یک لحظه زمان بصورت ایستا نشان می دهند و در مقابل مدل های پویا گذر زمان را نیز نشان می دهند و وضعیت سیستم را نسبت به زمان به نمایش می گذارند.

در مدل های قطعی کلیه تغییرات در مدل معین و براساس روابط غیراحتمالی صورت می گیرد و در مدل های احتمالی برخی از تغییرات یا روابط در مدل براساس روابط احتمالی یا تصادفی صورت می گیرد.

اغلب روش بررسی و تحلیل سیستم و نوع مدل مورد استفاده براساس محیط خود سیستم تعیین می شود.

Year. Month. Date. ( )

مدل های ریاضی تحلیلی در سیستم هایی که استفاده از این روش در آنها امکان پذیر باشد مطلوب ترین و دقیق ترین روش برای مطالعه سیستم است اما در سیستم هایی که به علت پیچیدگی زیاد و یا نیاز به نمایش واقعی تر رفتار سیستم استفاده از مدل های تحلیلی ریاضی امکان پذیر نباشد از شبیه سازی برای مطالعه سیستم استفاده می نمائیم.

انواع روشهای شبیه سازی:

۱- شبیه سازی همانی:

در این روش یک مدل همانند سیستم واقعی تولید می گردد و مطالعه بر روی آن مدل صورت می گیرد. گرچه این مدل ساده به نظر می رسد و نتایجی که از این روش بدست می آید ۱۰۰٪ قابل قبول است ولی معایبی نیز دارد.

معایب شبیه سازی همانی:

۱- غیر قابل استفاده بودن در بعضی از سیستم ها

۲- مخارج زیاد برای استفاده از آن

۳- کمبود امکانات کنترل متغیرهای سیستم

۴- غیر قابل استفاده بودن در مطالعه سیستم هایی که احتیاج به بررسی سریع و کسب نتایج در یک مدت کوتاهی دارند.

۲- شبیه سازی نیمه همانی

همانطور که از نام این روش برمی آید در این شبیه سازی سعی می شود تا آنجائیکه امکان دارد از اشیاء و قوانین واقعی سیستم استفاده شود. و فقط اشیاء و قوانینی که شبیه سازی همانی آنها امکان پذیر نباشد را با سمبل ها یا نمادهایی جایگزین می کنند یعنی بخشی از سیستم واقعی و بخشی غیر واقعی یا مجازی می باشد.

مثال: بازی تدارکاتی فوتبال یک مثال نیمه همانی به حساب می آید.
مانور تقاضی یک مثال نیمه همانی است.

۳- شبیه سازی آزمایشگاهی :
در این روش بعضی از اشیاء سیستم به وسیله امکانات آزمایشگاهی در مقیاس کوچکتر ساخته می شوند و بعضی از نمادها و روابط با سمبل هایی جایگزین می گردد. سپس مدل ساخته شده بوسیله عملگرهای واقعی سیستم در آزمایشگاه مورد بررسی قرار می گیرد بطور مثال بررسی یک هواپیما در تونل باد یک نوع شبیه سازی آزمایشگاهی است.

۴- شبیه سازی کامپیوتری :
در این روش مدل از سیستم بصورت یک برنامه کامپیوتری ساخته می شود. کلیه اشیاء و نمادهای سیستم بصورت ساختارهای داده ای و مشخصات و رفتار آنها بصورت توابع برنامه سازی پیاده سازی می گردد. این روش بدلیل اینکه در مطالعه اغلب سیستم ها مفیدی باشد کاربرد زیادی دارد.

خواص شبیه سازی کامپیوتری :

۱- قدرت فشرده سازی زمان :
در شبیه سازی کامپیوتری می توان چندین سال فعالیت یک سیستم را در چند ثانیه مشاهده و بررسی کنیم.

۲- قدرت گسترش زمان :
بوسیله شبیه سازی کامپیوتری تحلیلگر سیستم می تواند جزئیات تغییراتی را که در زمان واقعی قابل مشاهده نیستند را مشاهده و بررسی نماید. یا به عبارت دیگر تغییراتی که به علت بالا بودن سرعت وقوع آنها در سیستم واقعی قابل مشاهده نیستند در شبیه سازی کامپیوتری قابل کنترل و بررسی می باشد.

۳- قدرت توقف زمان :
در یک بررسی گاهی لازم است که حرکت زمان را متوقف نمائیم و نتایج بدست آمده تا آن لحظه را مطالعه کنیم و پس از اخذ تصمیمات لازم بررسی را از همان نقطه توقف ادامه دهیم.
لازمه این مطلب این است که کلیه پدیده های وابسته به سیستم و وضعیت خود را تا شروع مجدد بررسی حفظ نماید. که فقط در شبیه سازی کامپیوتری این امکان وجود دارد.

۴- امکان تکرار :
شبیه سازی کامپیوتری این امکان را به تحلیلگر سیستم می دهد که یک آزمایش یا بررسی را با حفظ کلیه شرایط

Year. Month. Date. ( )

اولیه سیستم چندین بار تکرار کند و در هر یک از دفعات تکرار بعضی از پارامترها را تغییر دهد و تاثیر آنرا بر روی رفتار سیستم مشاهده نماید.

۵- امکان بررسی سیستم های جدیدی که وجود خارجی ندارند:
شبیه سازی کامپیوتری علاوه بر اینکه می تواند در بررسی سیستم های موجود استفاده شود می تواند در بررسی سیستم هایی که در مرحله طراحی می باشند و وجود خارجی ندارند نیز مورد استفاده قرار گیرد قبل از صرف هرگونه نیرو، سرمایه، زمان برای ایجاد فیزیکی سیستم ها.

۶- شبیه سازی کامپیوتری برای مطالعه سیستم هایی که مطالعه آنها به روشهای دیگر خطرناک یا غیر ممکن است قابل استفاده می باشد.

مراحل شبیه سازی کامپیوتری:

۱- تعیین هدف و تعریف دقیق سیستم:

اولین مرحله از هر آزمایش از جمله شبیه سازی تعیین هدف و تعریف دقیق آزمایش مورد نظر است زیرا این هدف است که چگونگی آزمایش، جزئیات لازم و نتایج زمان را تعیین می کند.
پس از تعیین هدف باید به شناخت سیستم بپردازیم. و قسمت هایی که بطور مستقیم یا غیر مستقیم با هدف بستگی دارند مشخص کنیم.
منظور از تعریف دقیق سیستم مشخص کردن زیر سیستم ها، اشیاء، عوامل داخلی و خارجی و بالاخره پارامترها و متغیرهای سیستم است. بعد از تعریف دقیق بخش ها و اطلاعات فوق روابط و قوانین حاکم در سیستم که در ارتباط با هدف شبیه سازی هستند مشخص و فرموله می گردند.
و یک الگوریتم رفتاری از سیستم جهت شبیه سازی تولید می شود.

جلسه چهارم

۲- جمع آوری مشاهدات و تعیین تعداد پارامترها:

در برنامه‌های شبیه‌سازی باید مقدار پارامترها و توزیع آماری آنها بجز پارامترهای تصمیم قبل از ساختن مدل (نوشتن برنامه) تعیین یا تخمین زده شود. برخی از مواقع تعیین دقیق مقدار بعضی از پارامترها امکان پذیر نمی‌باشد و باید بصورت تخمین مقدار آنها مد نظر گرفت. برای تعیین مقدار پارامترها می‌توانیم از شمارش، اندازه‌گیری و یا رکوردهای موجود در سیستم استفاده کنیم.

برای پارامترهایی که تعیین دقیق آنها امکان پذیر نیست پس از نمونه برداری و جمع آوری تعدادی مشاهدات باید تخمین توزیع مناسب برروی آنها صورت گیرد. بطور مثال در یک سیستم بانک پارامترهایی نظیر فاصله زمانی بین ورود مشتریان، سرعت عمل کارمندان، میزان خرابی دستگاهها، تعیین یا تخمین زده شود.

۳- مدل سازی:

در این مرحله یک مدل از سیستم بصورت یک برنامه کامپیوتری نوشته می‌شود. عملیاتی که به این مرحله صورت می‌گیرد می‌توان به دو بخش تقسیم نمود:

۱- نوشتن الگوریتم و فلوچارت اجرای برنامه

۲- برنامه‌نویسی

در بخش اول با توجه به رفتار سیستم که در مرحله ۱ تعیین شده است الگوریتم یا فلوچارت برنامه تهیه می‌گردد. سپس با توجه به آن در بخش دوم برنامه کامپیوتری مدل نوشته می‌شود. برای برنامه‌سازی می‌توان از یکی از زبانهای برنامه‌نویسی عمومی نظیر C++، C#، VB و... استفاده کرد و یا از زبانهای مخصوص شبیه‌سازی نظیر GPSS یا GASP

۴- طرح آزمایش:

انجام یک شبیه‌سازی کاملاً مشابه انجام یک آزمایش در آزمایشگاه است. بعنوان مثال در آزمایشگاه مقادیر اولیه پارامترهای مؤثر بر آزمایش از قبیل درجه حرارت، فشار هوا، شدت جریان الکتریکی و... دارای اهمیت بوده و باید با یک طرح تعیین شده مدتی معلوم و اجرا گردند. در شبیه‌سازی نیز عواملی وجود دارند که باید دارای طرح معینی باشند نظیر شرایط اولیه، شرایط پایانی، شرایط زمانی که مدل باید اطلاعاتی را تولید

Year. Month. Date. ( )

کند و ....

آزمایش کننده باید قبل از انجام آزمایش شبیه‌سازی این عوامل را به روش‌های علمی و فنی مشخص کند. معمولاً در یک آزمایش شبیه‌سازی همه عوامل، قوانین و پارامترهای سیستم بجز یکی ثابت در نظر گرفته می‌شود و در هر بار آزمایش مقدار یکی از پارامترها را تغییر می‌دهیم تا میزان تأثیر آنرا بر روی سیستم بدست آوریم.

۵- احراز اعتبار:

این مرحله از مهمترین و مشکل‌ترین مراحل شبیه‌سازی است. احراز اعتبار یک مدل یعنی آنکه مدل ساخته شده تا چه حد رفتار سیستم واقعی را به درستی شبیه‌سازی می‌نماید. برای این منظور ابتدا باید ضوابط و معیارهای اندازه‌گیری میزان تطابق رفتار مدل با رفتار سیستم را مشخص کنیم سپس این معیارها را در مدل ساخته شده بررسی نماییم معمولاً دو روش برای اندازه‌گیری میزان اعتبار مدل‌های شبیه‌سازی بکار می‌رود:

۱- در مواردی که ارقام و نتایج سیستم واقعی در دسترس باشد این نتایج را با نتایج بدست آمده از مدل مقایسه می‌کنیم.

۲- هنگامی که سیستم واقعی موجود نباشد و یا ارقام یا نتایجی از رفتار سیستم واقعی در دسترس نباشد باید این مقادیر را محاسبه یا پیش‌بینی کنیم، سپس مقادیر مدل شبیه‌سازی را با آنها مقایسه نماییم.

۶- تجزیه و تحلیل نتایج:

آخرین مرحله از شبیه‌سازی تجزیه و تحلیل نتایج بدست آمده از مدل شبیه‌سازی است. همانطور که گفته شد هدف از شبیه‌سازی شناخت بهتر سیستم و بهبود در زمان نتایج آن در رابطه سیستم (اشکال زدایی) است. بنابراین آزمایش شبیه‌سازی باید به یکسری سئوال پاسخ دهد. انتخاب نتایج برنامه‌ی شبیه‌سازی بصورت تعدادی عدد است که باید بدقت تجزیه و تحلیل شوند تا وقتی آنها مشخص گردد و با استفاده از آنها به سئوالات تعیین شده پاسخ داده شود.

شبیه سازی کامپیوتری سیستم ها:

شبیه سازی کامپیوتری به ۲ روش امکان پذیر است:

۱- شبیه سازی به روش زمانبندی پیشامدها

۲- شبیه سازی به روش پویش فعالیت ها

در روش اول پیشامدهای سیستم تعیین می گردد. منظور از پیشامد زمانهایی است که تغییراتی در وضعیت سیستم رخ می دهد. سپس عملیاتی که در هنگام وقوع یک پیشامد رخ می دهد بصورت یک تابع پیاده سازی می گردد. سپس زمان شبیه سازی از لحظه شروع شبیه سازی هر بار افزوده می شود تا به زمان وقوع پیشامد در سیستم برسد. سپس پیشامد مورد نظر پردازش می گردد و این مراحل آنقدر ادامه می یابد تا به زمان پایان شبیه سازی برسد.

افزایش زمان به دو صورت امکان پذیر است:

۱- افزایش زمان با مقدار ثابت

۲- افزایش زمان با مقدار متغیر

گام یا
افزایش زمان با مقدار ثابت:

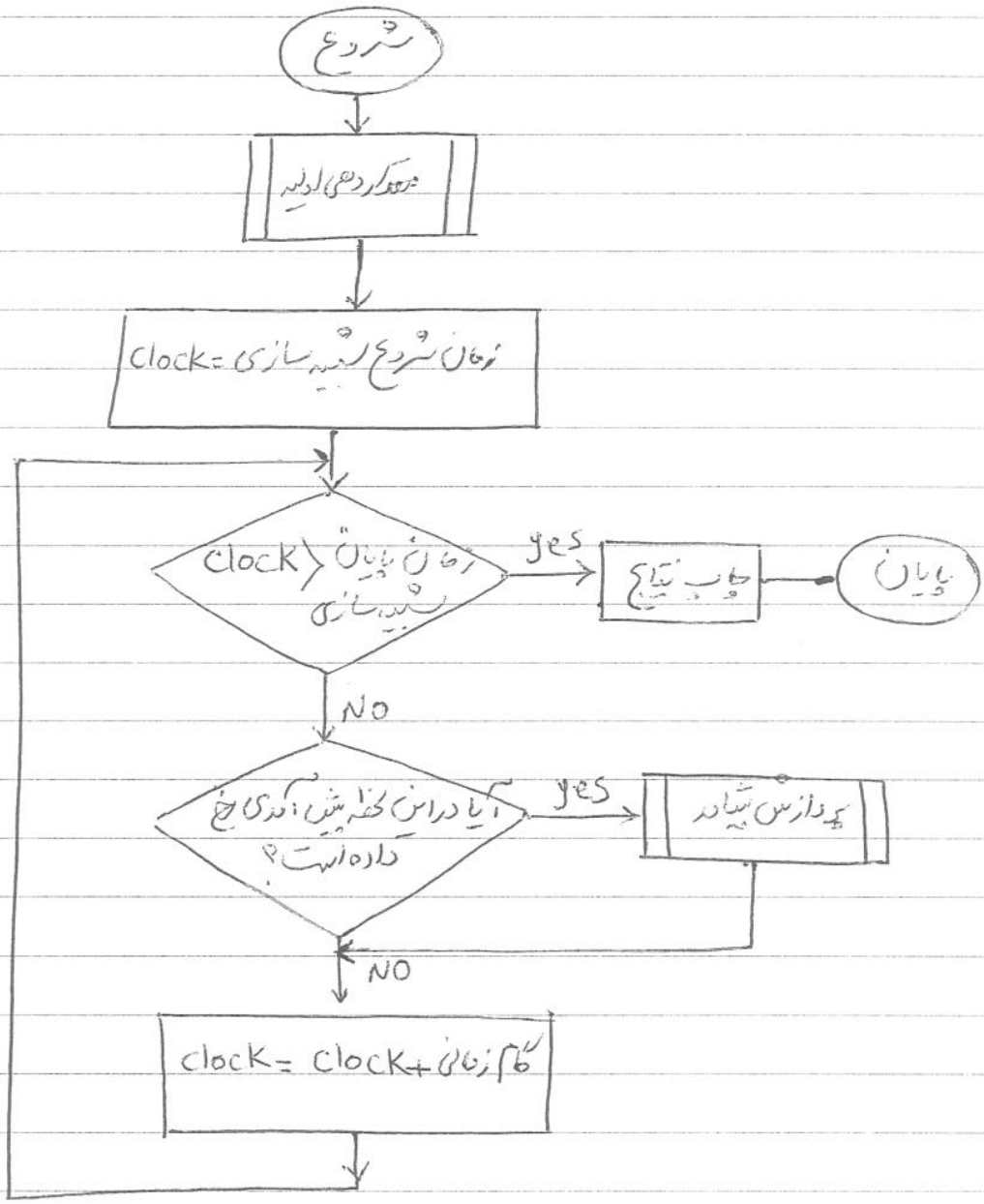
در این روش زمان شبیه سازی هر بار به اندازه یک مقدار ثابت که به آن گام زمانی یا time slice گفته می شود افزایش می یابد و در هر مرحله بررسی می کنیم که آیا در سیستم پیشامدی رخ می دهد و در صورت وقوع پیشامد تغییرات و عملیات مشخص شده را پردازش می کند. سپس یک گام زمانی دیگر برداشته می شود. و این مراحل آنقدر ادامه می یابد تا به زمان پایان شبیه سازی برسیم.

Year. Month. Date. ( )

نحوه تعیین گام زمانی:

مقدار گام زمانی به فواصل زمانی بین وقوع پیشامدها و تغییرات در سیستم بستگی دارد. اغلب در سیستم‌های گسسته این مقدار ترازی است با حداقل فاصله زمانی بین وقوع پیشامدها در نظر گرفته می‌شود. در سیستم‌های پیوسته که در آن‌ها دائماً تغییرات صورت می‌گیرد گام زمانی با توجه به دقت شبیه‌سازی در نظر گرفته می‌شود.

فلوچارت شبیه‌سازی به این روش بصورت زیر است:



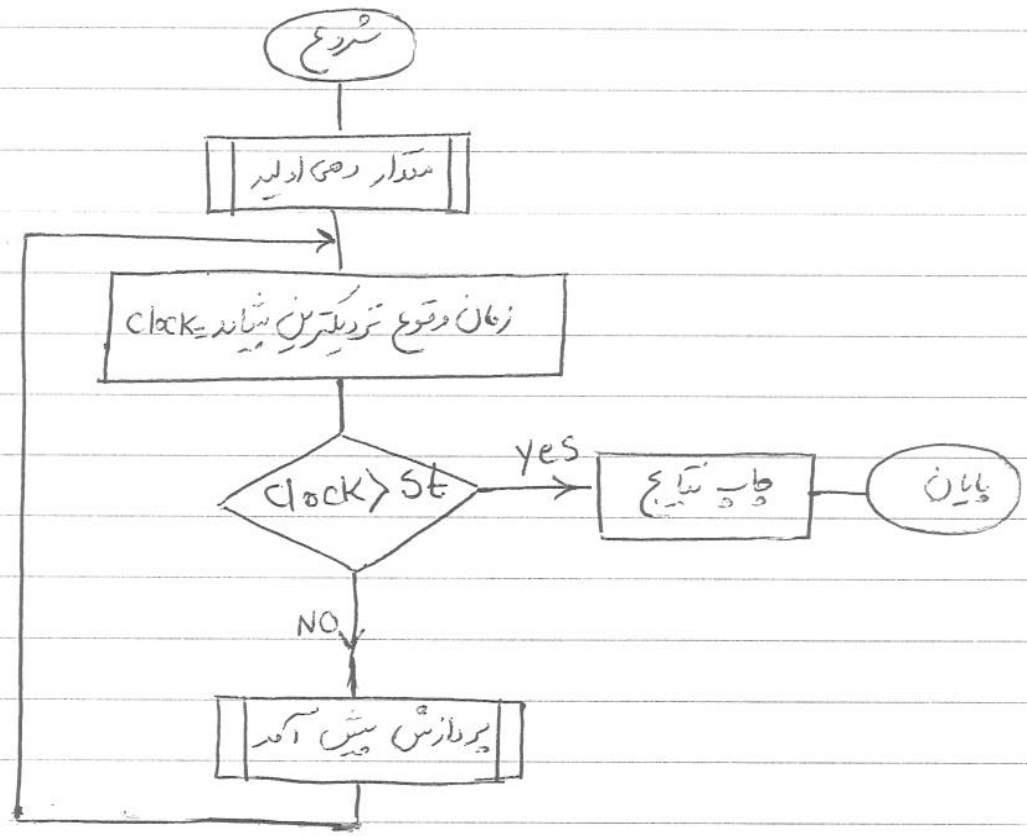
جلسه پنجم

روش دوم افزایش زمان با گام متغیر:

در روش قبلی (افزایش زمان با گام ثابت) مواردی پیش می آید که یک گام برداشته می شود و سیستم بررسی می شود که آیا پیشامدی در این لحظه رخ داده است یا خیر و هیچ پیشامدی رخ ندارد باشد. که بررسی سیستم به منظور اینکه پیشامدی رخ داده یا خیر ممکن است نیاز به عملیات طولانی داشته باشد و در اینگونه موارد جزء اتلاف وقت ثمره ای در بر نخواهد داشت.

در روش افزایش زمان با گام متغیر زمان شبیه سازی بصورت جهشی افزایش می یابد. تا از اینگونه اتلاف وقت جلوگیری گردد. یعنی در هر مرحله زمان وقوع پیشامد بعدی در سیستم پیش بینی می کنیم و ساعت شبیه سازی را به زمان وقوع پیشامد بعدی افزایش می دهیم.

فلوچارت این روش افزایش زمان بصورت زیر است:



St: زمان پایان شبیه سازی

Year. Month. Date. ( )

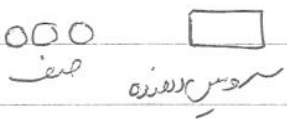
شبیه سازی سیستم های صفی :

بررسی سیستم های مختلف نشان می دهد که در بسیاری از سیستم ها، اشیایی وارد سیستم می شوند و تقاضای انجام یک یا چند سرویس را از سرویس دهنده های سیستم می نمایند و در صورتیکه سرویس دهنده ها اشغال باشند در یک صف انتظار قرار می گیرند به اینگونه سیستم ها، سیستم های صفی گفته می شود. سیستم های نظیر بانک، فروشگاه، تعمیرگاه، فرودگاه، انبار، بیمارستان، سیستم های کامپیوتری نظیر سیستم عامل، پمپ بنزین و ... نمونه هایی از سیستم های صفی می باشند.

انواع سیستم های صفی :

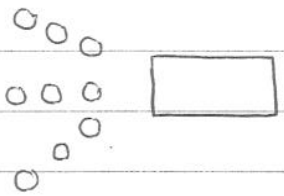
با توجه به تنوع سرویس های مورد تقاضای اشیاء و تعداد سرویس دهندگان در سیستم انواع سیستم های صفی بوجود می آیند که عبارتند از :

1. سیستم یک صف یک سرویس دهنده



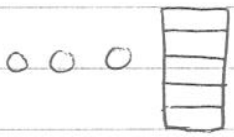
نظیر مطب پزشک، آبدارخانه

2. چند صف یک سرویس دهنده



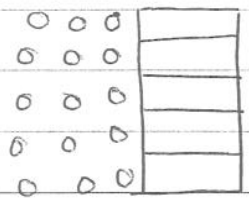
صف نانوایی، مثل سیستم مدیریت پردازنده سیستم عامل سیستم زمانبندی پردازنده که job ها در صف های مختلف قرار می گیرند.

3. یک صف چند سرویس دهنده



نظیر صف بانک

4. چند صف چند سرویس دهنده موازی



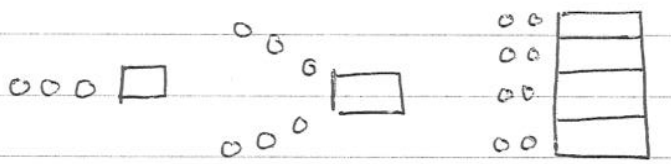
مانند عوارض

5- چند صف چند سرویس دهنده موازی

مانند خط تولید اتومبیل

6- ترکیبی از چند صف چند سرویس دهنده

تعاریف سیستم انتخاب واحد دانشگاه



شبیه سازی سیستم یک صف یک سرویس دهنده

این سیستم ساده ترین نوع سیستم صفی محسوب می شود. در این سیستم اشیایی که اصطلاحاً به آنها مشتری می گوییم وارد سیستم شده و تقاضای یک سرویس یا از تنها سرویس دهنده سیستم می نمایند در صورتی که سرویس دهنده آزاد باشد بلافاصله سرویس به آن شی داده شروع می کند در غیر اینصورت اشیاء در یک صف انتظار قرار می گیرند تا زمانی که سرویس دهنده آزاد شود و سرویس به وی امکان پذیر گردد. می خواهیم برنامه‌ای بنویسیم که وضعیت این سیستم را در هر لحظه نمایش دهد. وضعیت سیستم به وسیله اشیایی که در صف قرار دارند طول صف و وضعیت سرویس دهنده (آزاد یا مشغول بودن) مشخص می گردد.

برای شبیه سازی سیستم ابتدا باید پیش آمدهایی که باعث تغییر وضعیت سیستم می شود را شناسایی کنیم.

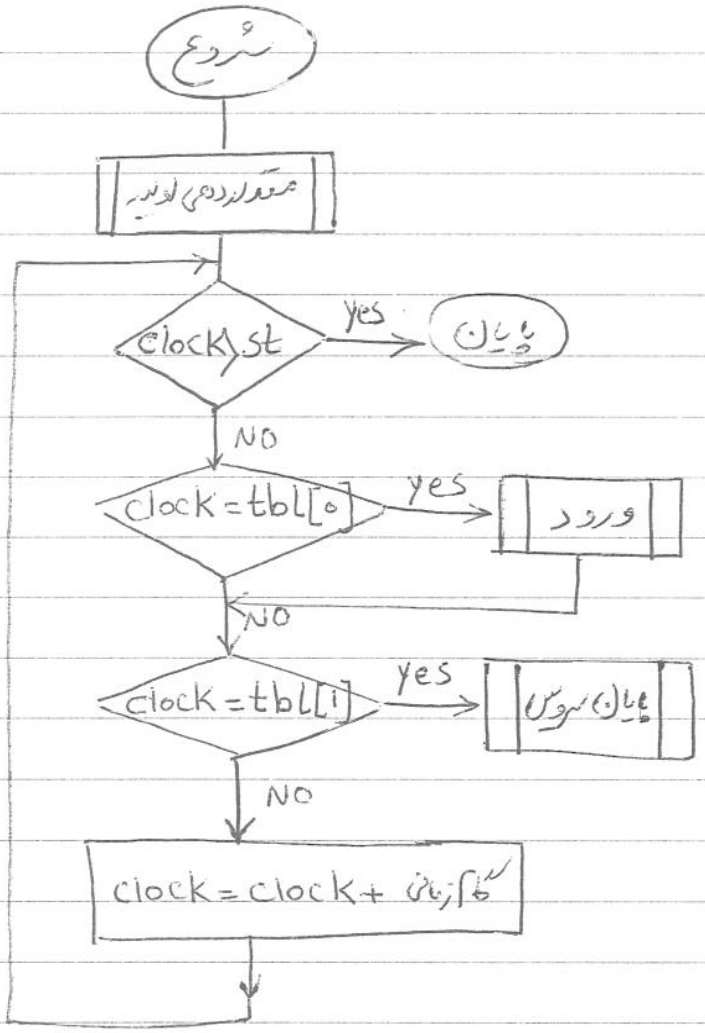
در این سیستم ۲ نوع پیشامد رخ می دهد:

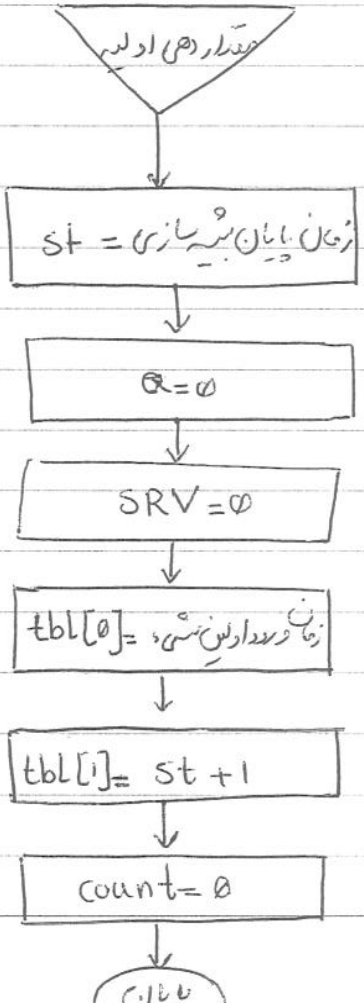
1- پیشامد ورود

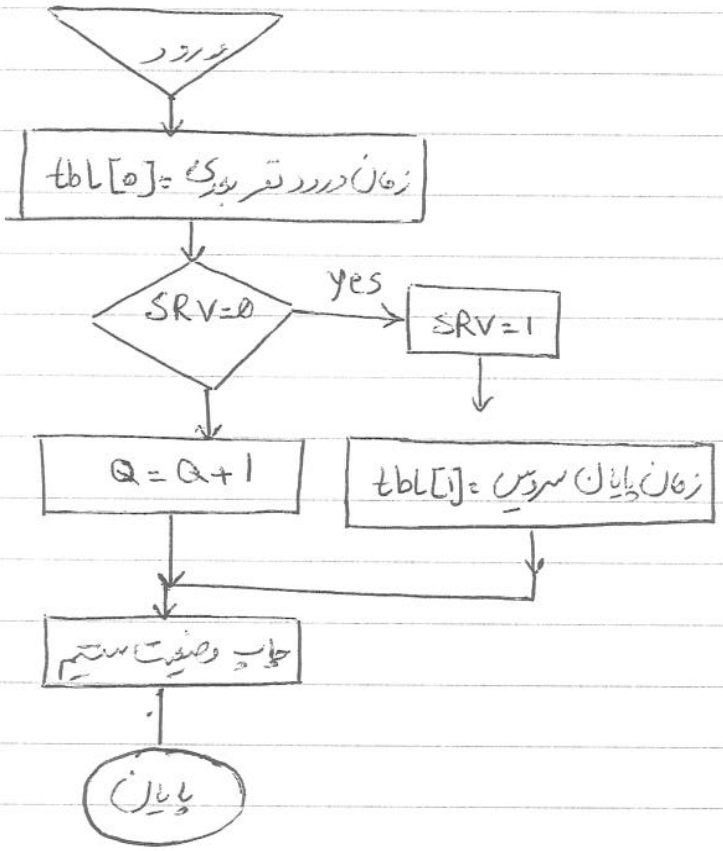
2- پیشامد پایان سرویس (خروج یک مشتری)

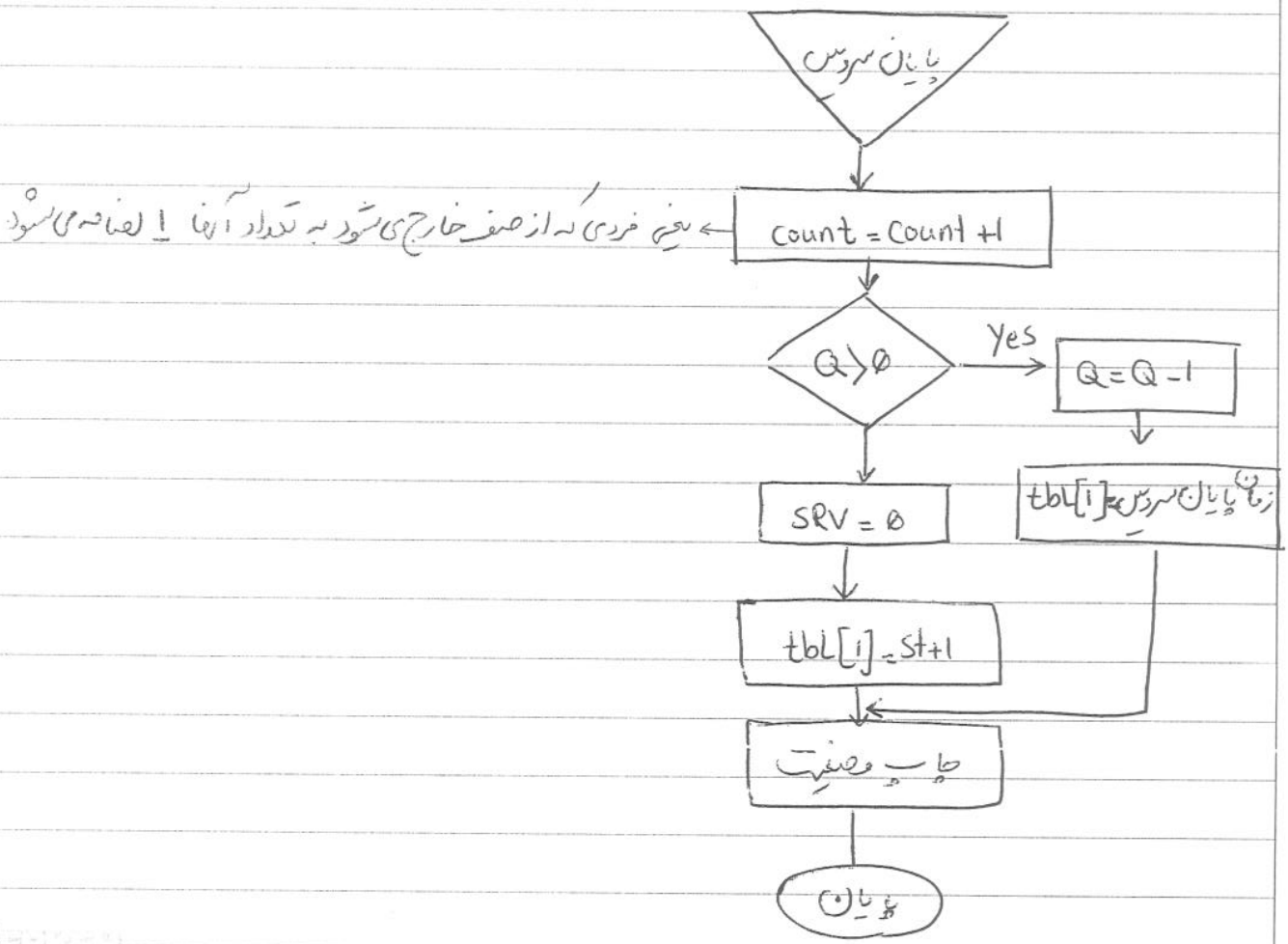
فلوچارت شبیه سازی این سیستم به روش افزایش زمان با گام ثابت بصورت زیر می باشد.

Year. Month. Date. ( )






ورود
زمان ورود تقریبی = tbl[a]
SRV=0
yes
SRV=1
Q=Q+1
زمان پایان سرویس = tbl[i]
چاپ وضعیت سیستم
پایان


پایان سرویس
count = count +1
یعنی فردی که از صف خارج می‌شود به تعداد آنها 1 اضافه می‌شود
Q>0
yes
Q=Q-1
SRV = 0
زمان پایان سرویس = tbl[i]
tbl[i] = St+1
چاپ وضعیت
پایان

Year. Month. Date. ( )

شرح پارامترهای سیستم:

St: زمان پایان شبیه سازی

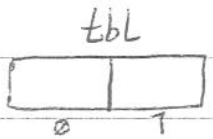
clock: ساعت شبیه سازی

Q: طول صف یا تعداد اشیاء داخل صف انتظار

SRV: وضعیت سرویس دهنده (صفر یعنی آزاد - یک یعنی مشغول)

count: تعداد اشیایی که از سیستم خارج شده اند.

آرایه tbl[]: زمان وقوع پیشامدهای بعدی را نشان می دهد. چون در این سیستم دو نوع پیشامد داریم آرایه tbl[] دارای ۲ خانه می باشد. tbl[0] زمان وقوع پیشامد ورود بعدی و tbl[1] زمان وقوع پیشامد پایان سرویس بعدی را نشان می دهد



همانطور که شروع در مقداردهی اولیه زمان ورود نفر اول را می توانیم تخمین بزنیم بنابراین tbl[0] را مقداری می دهیم ولی چون زمان پایان سرویس مشخص نمی باشد tbl[1] را مقداری خارج از زمان شبیه سازی قرار می دهیم. همچنین هرگاه در سیستم یک نفر وارد سیستم می شود می توانیم زمان ورود نفر بعدی را تخمین بزنیم و همانگاه که سرویس دهنده شروع به انجام سرویس یک مشتری می کند می توانیم زمان پایان سرویس آنرا تخمین بزنیم و هرگاه سرویس دهنده آزاد باشد چون زمان پایان سرویس آن نامشخص است مقدار tbl[1] را St+1 قرار می دهیم.

سوال ۱۱:

اگر در یک سیستم یک صف یک سرویس دهنده فاصله زمانی بین ورود مشتریان به ترتیب اعداد 6، 4، 8، 9، 10، 12، 3، 4، 2، 5 باشد و مدت زمان انجام سرویس مشتریان به ترتیب 7، 6، 8، 3، 5، 2، 4، 3، 14، 10 باشد برنامه شبیه سازی این سیستم را به زبان C++ برای مدت زمان 60 واحد زمانی به روش اقتراسین زمان با گام ثابت بنویسید.

```
# include <iostream.h>

int v[]={5,2,4,3,12,10,9,8,4,6};

int p[]={10,14,3,4,2,5,3,8,6,7};

int tbl[2];

int clock,Q,SRV,count,st,i;        // i: اندیس آرایه v ، اندیس آرایه p

void init(), voroud(), payan();

---

main()
{
  init();
  while( clock <= st)              // st: زمان پایان شبیه سازی
   { if(clock==tbl[0])
       voroud();
     if( clock = = tbl[1])
       payan();
     clock = clock+1;
   }
}
```

Year. Month. Date. ( )

```
void init()
{
  clock = 0;
  st = 60;
  Q = 0;
  SRV = 0;
  count = 0;
  tbl[0] = v[0];
  i = 1;
  tbl[1] = st + 1;
  j = 0;
}

void voroud()
{
  tbl[0] = clock + v[i];          زمان ورود نفر بعدی
  i++;
  if (SRV == 0)
  {
    SRV = 1;
    tbl[1] = clock + p[j];        زمان پایان سرویس اولین نفر
    j++;                          و چون یک عدد از p برداشته شده به j 1 واحد اضافه شد
  }
  else
  Q = Q + 1;
  Cout << clock << Q << SRV << count << "voroud in";
```

```
void payan()
{
  count = count + 1;
  if(Q>0)          اگر کسی در صف باشد
  {
    Q = Q-1;       یکی از صف کم می‌شود
    tbl[j] = clock + p[j];
    j++;
  }
  else
  {
    SRV = 0;
    tbl[j] = st + st+1      چون زمان پایان مشخص است قرار می‌دهیم
  }
cout << clock << Q << SRV << cout << "payan \n";
```

آخرین فرمان سیستم یک صف یک سرویس دهنده را به صورتی تغییر دهید که اولاً از روش افزایش زمان بگام متغیر استفاده نماید ثانیاً در پایان شبیه‌سازی متوسط زمان انتظار، متوسط زمان پاسخ و راندمان سرویس دهنده را محاسبه و چاپ نماید.

متوسط زمان انتظار = تعداد کسانیکه کارشان شروع شده / مجموع زمان انتظار

متوسط پاسخ = تعداد کسانیکه کارشان تمام شده / مجموع زمان پاسخ

زمان انتظار = شروع − ورود

زمان پاسخ = خروج − ورود

راندمان سرویس دهنده = مجموع کار کرد سرویس دهنده / کل زمان شبیه‌سازی یا مجموع کار کرد سرویس دهنده / کل زمان شبیه‌سازی

جواب تمرین

```
#include<iostream.h>
Int v[]={...

Int p[]={...

Int tbl[2];
Int clock,Q,Srv,count,st,i,j;
int vr[100];
int start,sume,sump,sumk,k,code;

Void init(),voroud(),payan();
main()
{
  init();
  if(tbl[0]<tbl[1])
    {
      clock=tbl[0];
      code=1;
    }
  else
    {  clock=tbl[1];
       code=2;
    }
  while(clock<=st)
   {
    if(code==1)
      voroud();
    else
      payan();
    if(tbl[0]<tbl[1])
    {
```

```
    clock = tbl[0];
    code = 1;
  }
  else
  {
    clock = tbl[1];
    code = 2;
}}
```

```
  cout << sume / j << '\n';
  cout << sump / count << '\n';
  sumk = sumk + clock - start;
  cout << sumk / st;
}
```

مجموع زمانهای انتظار / تعداد کسانیکه کارشان شروع شده = متوسط زمان انتظار — چاپ

چاپ متوسط زمان پاسخ

آخرین مرحله از کار کرد سرویس دهنده با مجموع کارکردها قبلی جمع می‌کنیم.

زمان

```
void init()
{ Q=0;
  srv=0;
  count=0;
  st=60;
  tbl[0] = v[0];
  i=1;
  tbl[i] = st+1;
  j=0;
  sume = sump = sumk=0;
  k=0;
}
```

tbl

15	25
0	1

```
void voroud()
{
  tbl[0] = clock + v[i];
  i++;
  vr[k] = clock;
  k++;
```

ورود

Year. Month. Date. ( )

```
if (SRV==0)
{
  SRV=1;
  tbl[i] = clock+p[j];        شروع بکار
  j++;
  start = clock;
}
else
   q++;
   cout<< clock <<q<<SRV<<"voroud \n";
}
```

آرایه vr[] : زمان ورود اشیاء به سیستم را در آن قرار می‌دهیم.

start : زمان شروع به کار سرویس دهنده را نشان می‌دهد

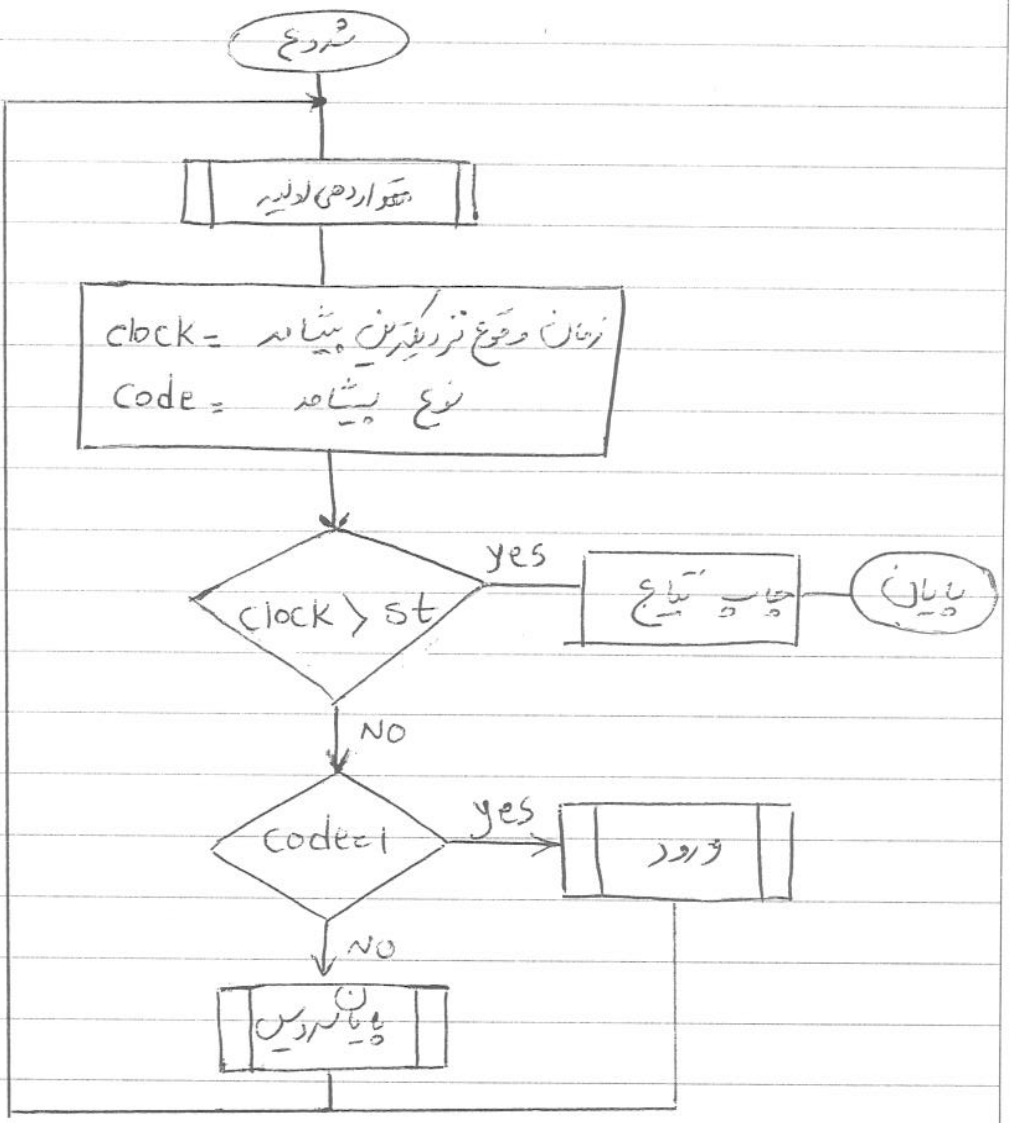
sume : مجموع زمان‌های انتظار

sump : مجموع زمان‌های پاسخ

somk : مجموع زمان کارکرد سرویس دهنده

k : شمارشگر آرایه vr[]

code : نوع پیشامد ۱ پیشامد ورود ۲ پایان سرویس




```
void payan()
{
  sump = sump + clock - vr[count];   زمان پاسخ
  count++;
  if (Q>0)
  {
    Q--;
    tbl[i] = clock + p[j];   شروع بکار
    sume = sume + clock - vr[j];   زمان انتظار
    j++;   شماره کارشان شروع شده
  }
  else
  {
    srv = 0;
    tbl[i] = st+1;   بیکار شدن سرویس دهنده
  }
```

Year. Month. Date. ( )

```
sumk = sumk + clock - start;
start = st+1;
}
cout << clock << Q << sRv << "payan\n";
```

زمان کارکرد

در لحظه ورود یک نفر زمان ورود توسط ی تعیین می گردد و در لحظه شروع به کار یکنفر زمان پایان کار آن تعیین می گردد هرگاه سرویس دهنده بیکار می گردد به علت اینکه زمان پایان کارش نامشخص است زمان پایان کار سرویس دهنده را st+1 قرار می دهیم.

در لحظه پایان کار یکنفر زمان پاسخ آن تعیین می شود و زمان انتظار یکنفر لحظه خروج از صف و شروع به کارش تعیین می شود و مدت زمان کارکرد یک سرویس دهنده در لحظه ای که بیکار می شود و دست از کار می کشد محاسبه می گردد یعنی از زمان شروع به کارش تا زمان دست از کار کشیدنش.

مثال) در یک سیستم یک صف یک سرویس دهنده فاصله زمان بین ورود اشیاء به سیستم به ترتیب 4 و 8 و 7 و 12 و 14 و 5 و 2 و 3 و زمان انجام سرویس هریک از اشیاء به ترتیب این اعداد 6 و 7 و 5 و 6 و 2 و 3 و 9 و 8 و 10 می باشد متوسط زمان انتظار، متوسط زمان پاسخ و راندمان سرویس دهنده را با استفاده از جدول پیگیری بدست آورید. برای مدت زمان 50 واحد زمان.

ورود	شروع	پایان	انتظار	پاسخ	Sume	Sump	start	Sumk
					0	0		0
3	3	13	0	10	0	10	3	
5	13	21	13-5=8	21-5=16	8	26		
10	21	30	21-10=11	30-10=20	19	46		
24	30	33	6	9	25	55	st+1	33-3=30
36	36	38	0	2	25	57	36 st+1	38-36=2 30+2=32
43	43	47	0	4	25	61	43 st+1	47-43=4 32+4=36
51								

متوسط انتظار $= \frac{25}{6} = 4.16$

راندمان $= \frac{36}{50} = \%72$

متوسط پاسخ $\frac{61}{6} = 10.16$

جلسه هفتم

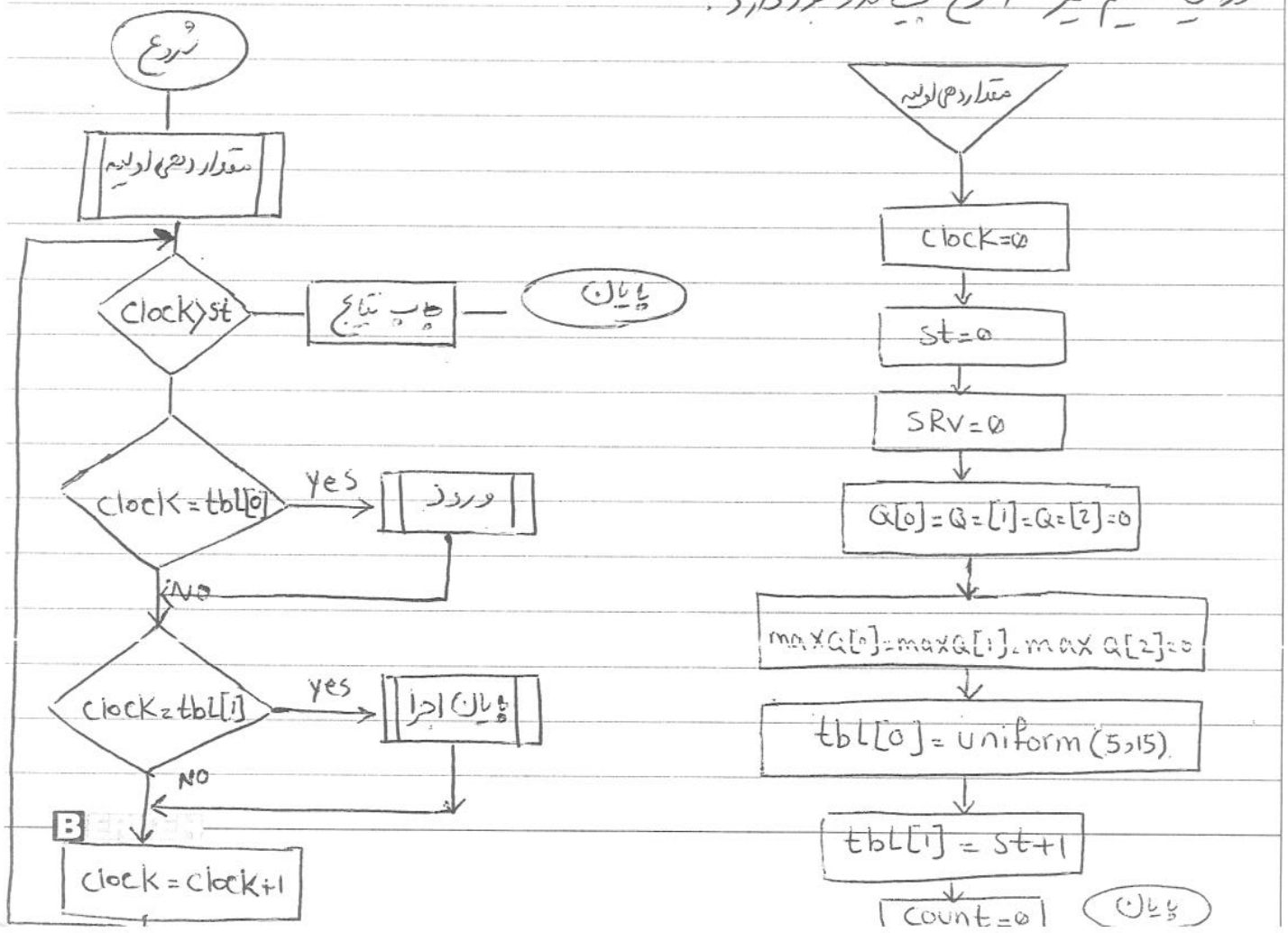
سیستم چند صف، یک سرویس دهنده:

در یک سیستم کامپیوتری در هر لحظه فقط می تواند یک برنامه را اجرا نماید. برنامه هایی که جهت اجرا وارد یک سیستم می شوند به سه نوع a و b و c تقسیم می گردند. برنامه های نوع A سبقت به نوع b و نوع b سبقت به نوع c از تقدم بالاتری برخوردارند.

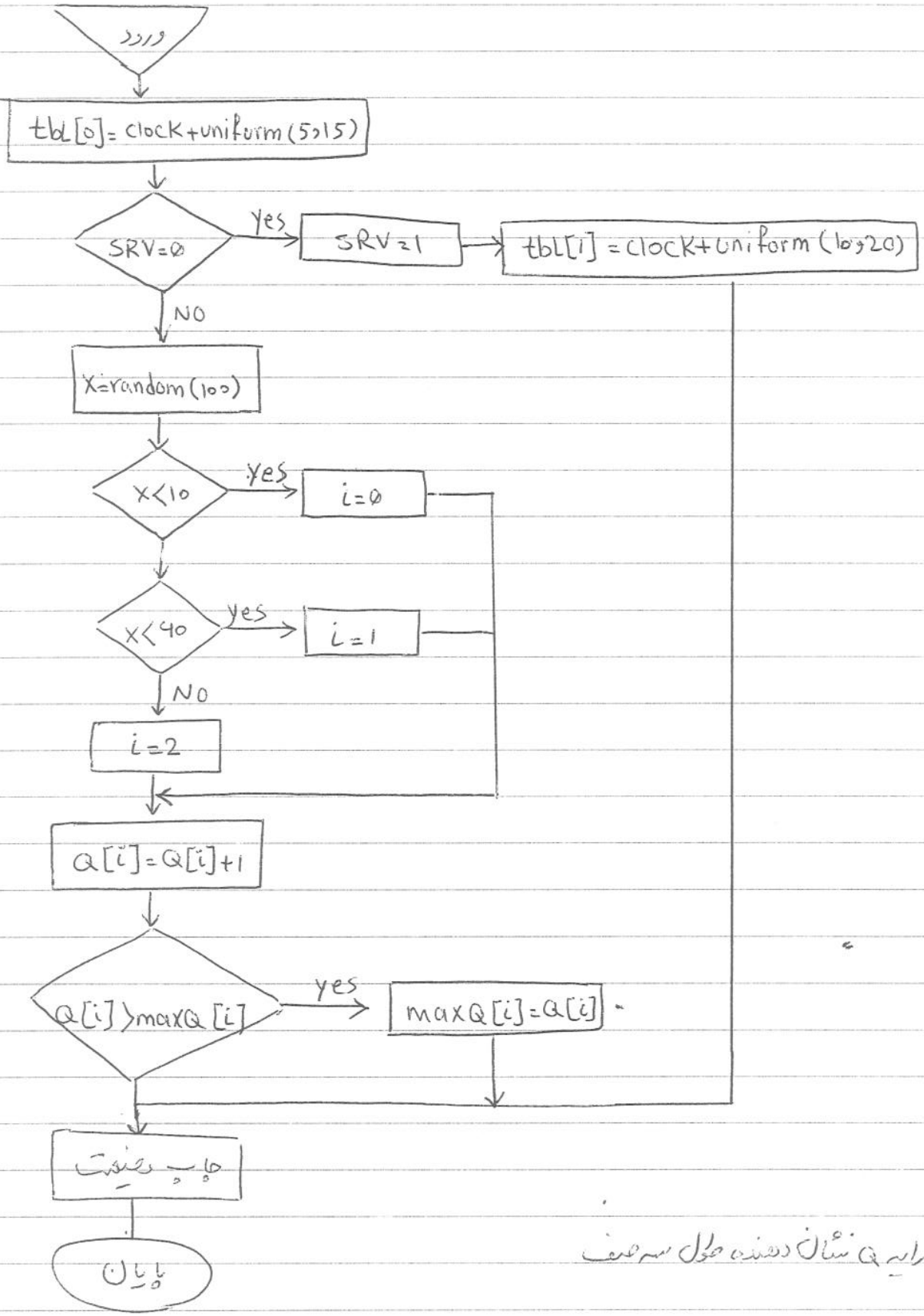
بنابراین در صف های جداگانه قرار می گیرند. اگر فاصله زمانی بین ورود برنامه ها به سیستم دارای توزیع یکنواخت در بازه [5و15] باشند و زمان اجرای برنامه ها نیز بطور یکنواخت بین ۱۰ تا ۲۰ واحد زمانی طول بکشد و ۱۰ درصد برنامه ها از نوع a و ۳۰ درصد از نوع b و ۶۰ درصد از نوع c باشند. برنامه شبیه سازی این سیستم را با استفاده از روش افزایش زمان با گام ثابت برای مدت زمان ۵۰۰ واحد زمانی بنویسیم به طوریکه وضعیت سیستم را در هر لحظه نمایش دهد و در انتها حداکثر طول هر یک از صف ها را چاپ نماید.

فلوچارت شبیه سازی این سیستم بصورت زیر است:

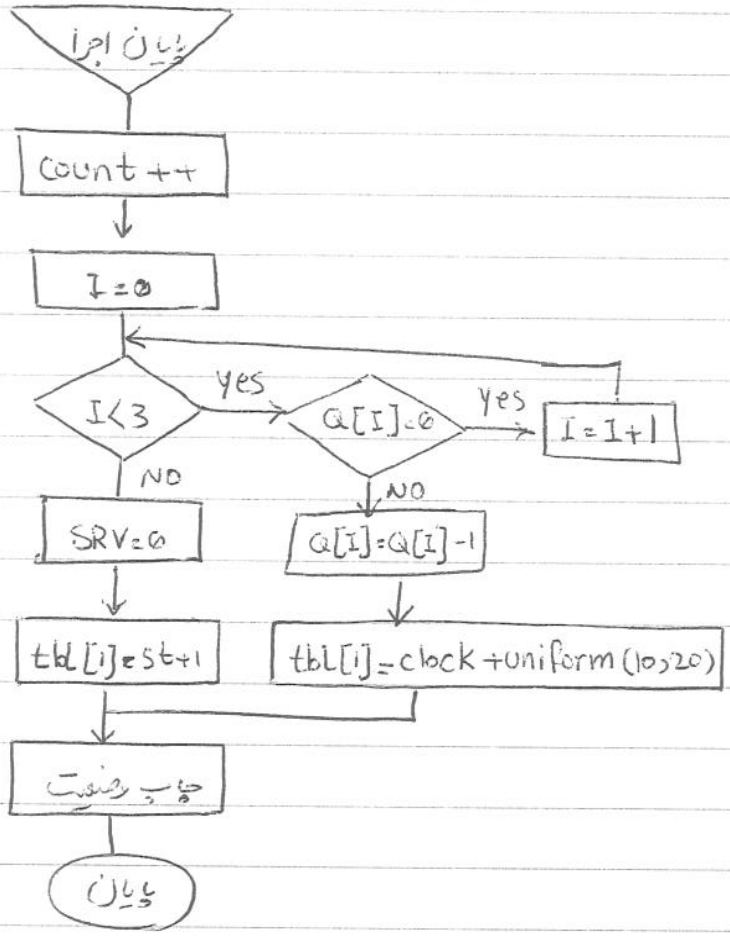
در این سیستم نیز ۲ نوع پیشامد وجود دارد:



Year. Month. Date. ( )



آرایه Q نشان دهنده طول سه صف




```
# include <iostream.h>
  int tbl[2];
  int Q[3], maxQ[3];
  int clock, st, SRV, count;
  void init(), voroud(), payan();
  Init uniform (int a, intb)
```

uniform تابع

```
  main()
  { init();
    while( clock<=st)
     { if (clock == tbl [0])
          voroud();
       if (clock == tbl[1])
          payan();
       clock++;
     }
    cout<<maxQ[0]<<maxQ[1]<<maxQ[2];
  }
```

Year. Month. Date. ( )

```
int uniform (int a, int b)
{ int x;
  x=random (100);
  return (a+(b-a)*x/100);
}
void init()
{
 st= 500
 clock=0;
 Count=0;
 Q[0]=Q[1]=Q[2]=0;
 maxQ[0]=maxQ[1]=maxQ[2]=0;
 SRV=0;
 tbl[0]=uniform (5,15);
 tbl[1]= st+1;
}

void voroud()
{
 int x,i;
 tbl[0]=clock+uniform(5,15);
 if(SRV==0)
 {
   SRV=1;
   tbl[1]= clock+uniform(10,20);
 }
 else
 { x=random(100);
   if (x<10) i=0;
   else if (x<40) i=1;
   else i=2;
```

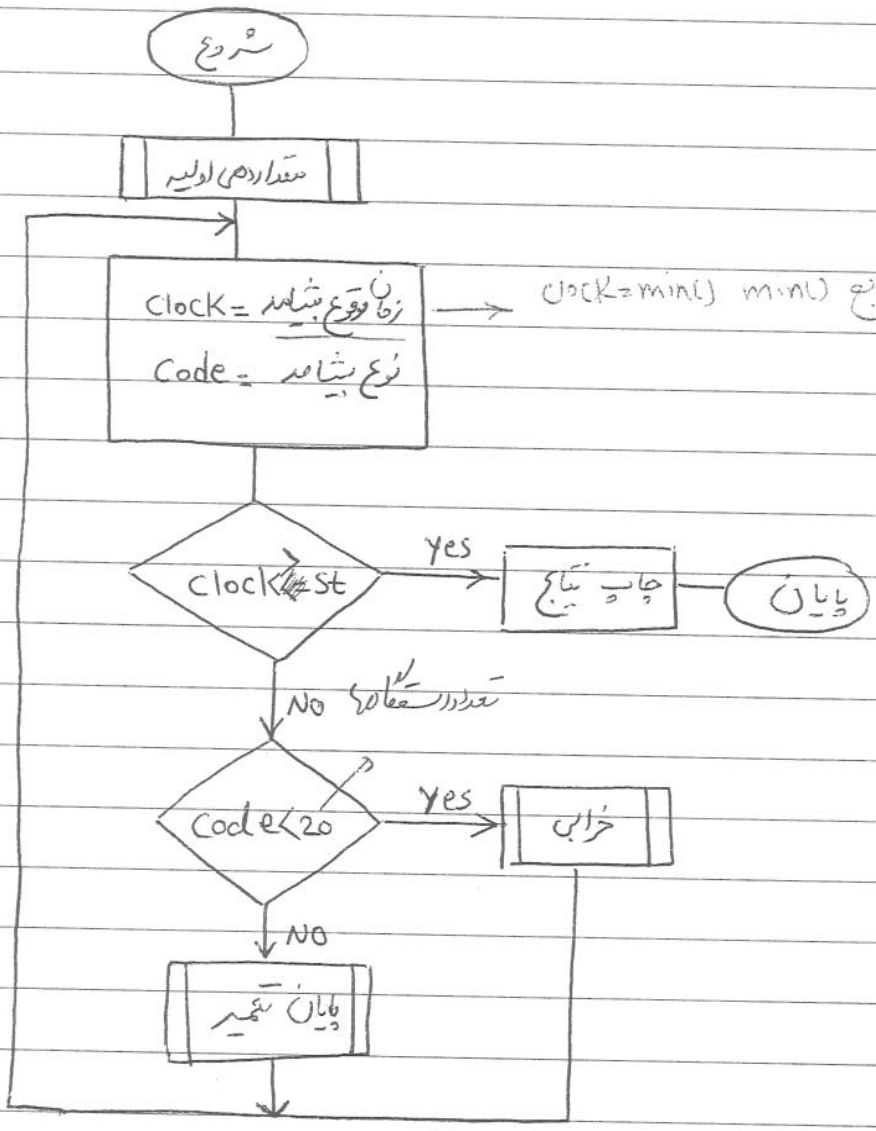
```
Q[i] = Q[i]+1;
if (Q[i]>maxQ[i])
    maxQ[i]=Q[i];
}
cout<< clock<< SRV<<Q[0]<<Q[1]<<Q[2]<<"voroud\n";


void payan()
{ int i;

  count=count+1;
  for(i=0; i<3; i++)
  if (Q[i]>0)
   {
    Q[i]=Q[i]-1;
   tbL[1]=clock+uniform(10,20);
    break;
   }
   if (i==3)
    {
     SRV=0;
     tbL[1]=St+1;
     cout<<clock<<SRV<<Q[0]<<Q[1]<<Q[2]<<"payan\n";
}
```

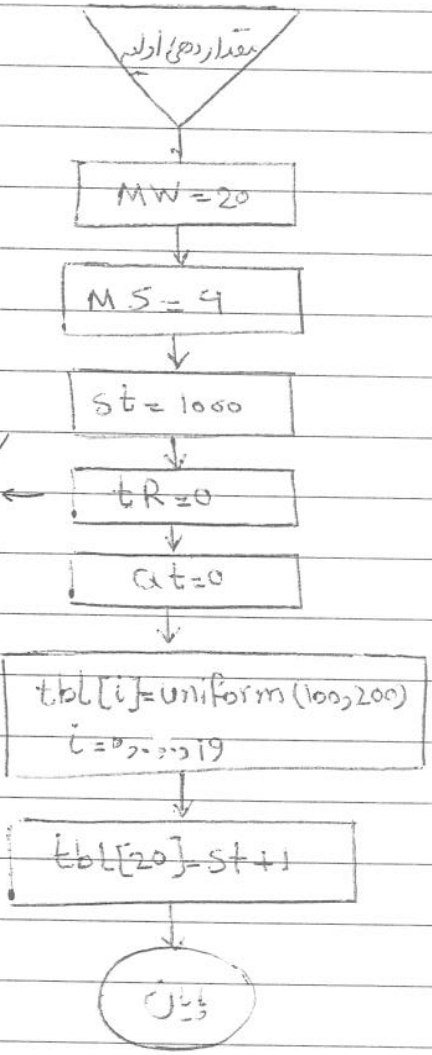
سیستم چند صف چند سرویس دهنده:

در یک کارگاه ساختمانی 20 دستگاه مشغول به کار می باشند و 6 دستگاه نیز بعنوان یدکی وجود دارد. دستگاهها بین از مدت زمانی بصورت تصادفی خراب می شوند. و در صورت وجود دستگاه یدکی از بین دستگاههای یدکی یکی جایگزین دستگاه خراب می گردد در غیر اینصورت جای آن خالی می ماند. دستگاههای خراب جهت تعمیر به قسمت تعمیرگاه فرستاده می شوند که در آن بخش یک تعمیرکار وجود دارد که در صورت آزاد بودن بلافاصله تعمیر دستگاه مورد نظر شروع می کند در غیر اینصورت دستگاه در یک صف انتظار قرار می گیرد تا تعمیرکار آزاد گردد. دستگاه تعمیر شده در صورت خالی بودن محلی در

Year. Month. Date. ( )

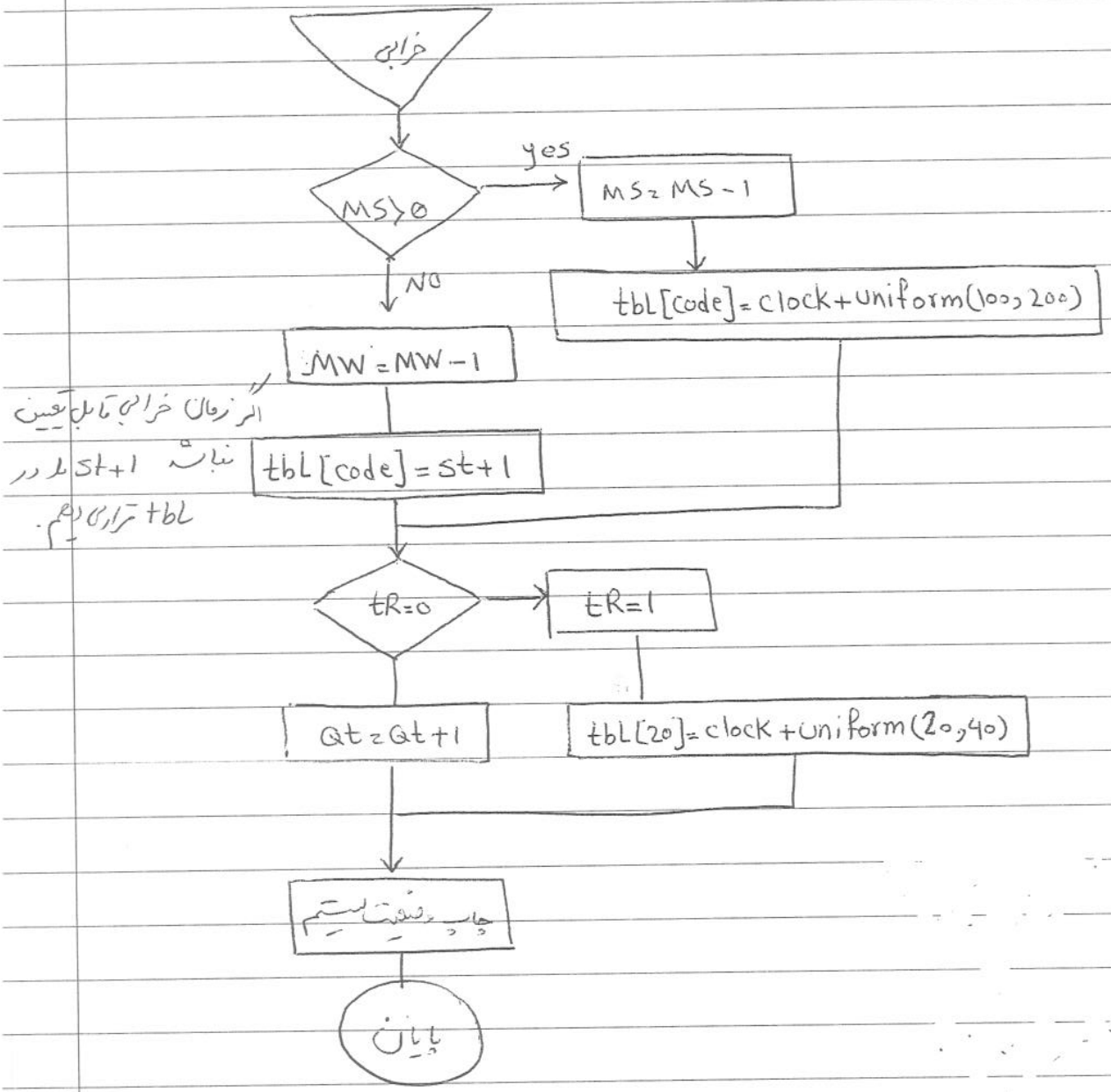
کارگاه در آن محل بکار گرفته می‌شود در غیر اینصورت به ماشین های بیکاری اضافه می‌گردد.
اگر فاصله زمان بین خرابی دستگاه‌ها بطور یکنواخت بین ۱۰۰ تا ۲۰۰ واحد زمانی
طول بکشد و زمان انجام تعمیر هر دستگاه نیز به طور یکنواخت بین ۲۰ تا ۶۰ واحد
زمانی طول بکشد برنامه شبیه سازی این سیستم را برای مدت زمان ۱۰۰۰ واحد
زمانی با گام متغیر شبیه سازی بطوریکه وضعیت سیستم را در هر لحظه نشان دهد.

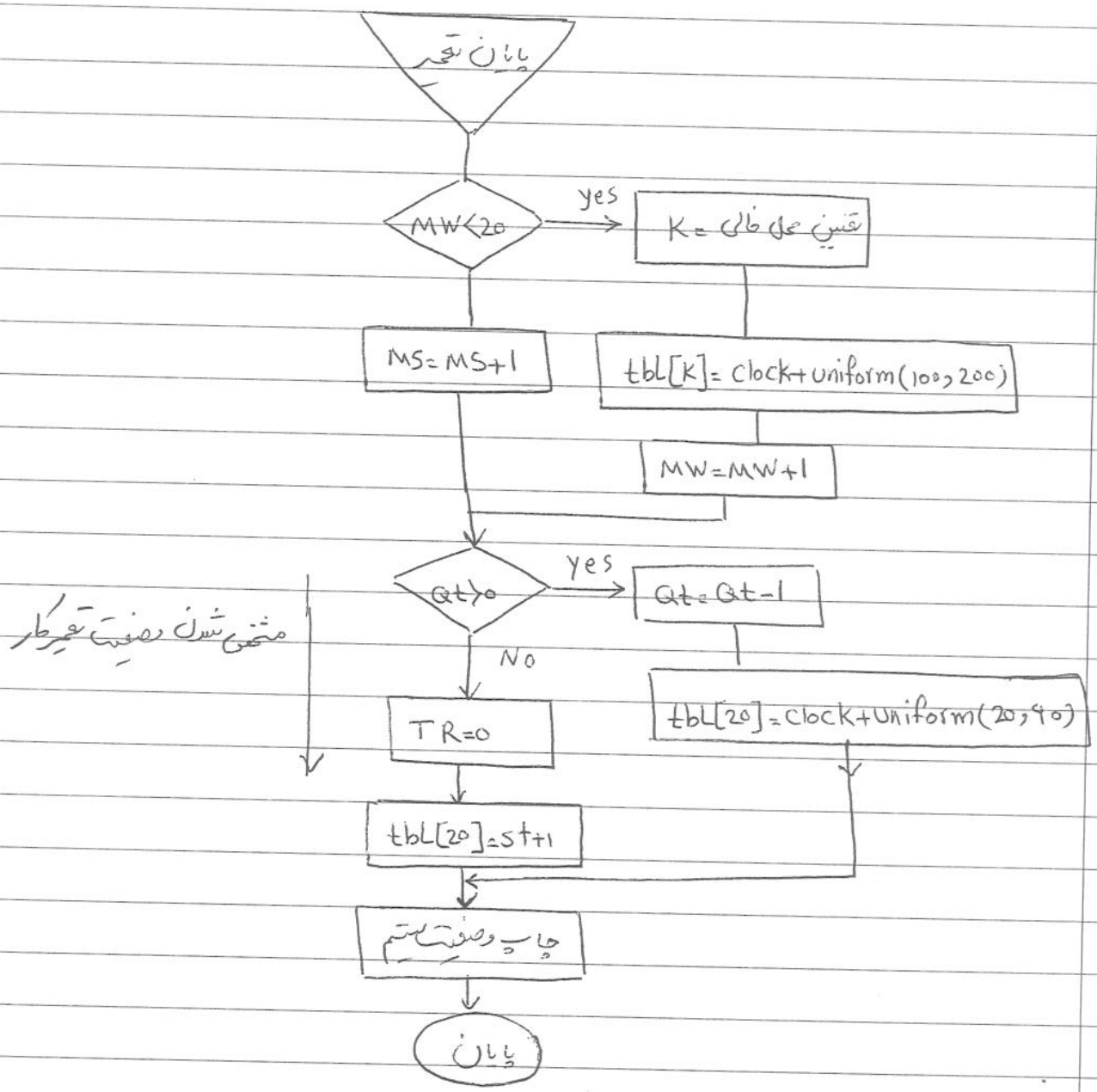




اگر تعداد تعمیرکارها ۲ تا بود tR[0 تا 1] = 0
۳ tR[0 تا 2] = 0

زمان خرابی ۲۰ تا ماشین را در tbl[i] به صورت یکنواخت تعیین می‌کنیم.
از صفر تا ۱۹
حلقه for
for(int i=0; i<20; i++)
tbl[i] = uniform
زمان پایان تعمیر


خرابی
MS>0
yes
MS = MS - 1
NO
tbl[code] = clock + uniform(100, 200)
MW = MW - 1
اگر زمان خرابی قابل تعیین نباشد st+1 را در tbl قرار می‌دهیم.
tbl[code] = st+1
tR=0
tR=1
Qt = Qt+1
tbl[20] = clock + uniform(20,40)
چاپ نتیجه لیستم
پایان



شرح پارامترهای سیستم:

MW: تعداد ماشین‌های معال

Qt: طول صف ماشین‌های منتظر تعمیر

MS: تعداد ماشین‌های یدک

St: زمان پایان شبیه‌سازی

tR: وضعیت تعمیرکار

در تابع مقداردهی اولیه زمان وقوع خرابی 20 ماشین را تعیین می‌کنیم و در tbl صفر تا 19 قرار می‌دهیم و زمان وقوع پایان تعمیر چون نامشخص است tbl 20 را یک عدد بزرگ مثل st+1 قرار می‌دهیم.

```
#include <iostream.h>
    int tbl[21];
    int clock, st, code, tR, qt, MS, MW;
    void init(), kharabi(), payantamir();
    int min();
    int uniform(int a, int b);
    main()
    {
     init();
     clock = min();
     while(clock<=st)
      {
       if(code<20)      kharabi();
       else             payantamir();
       clock = min();
      }
    }
     int min()
     {
      int m,i;
      m= tbl[0];
     for( i=0; i<21 ; i++)
        if ( tbl[i]<m )
          {
           code = i;
           m = tbl[i];
          }
        return(m);
     }
```

تابع تعیین زمان وقوع بیشاهدی

نوع بیشاهدی

این تابع حقیر مقدار آرایه tbl زمان وقوع بیشاهدی و شماره کوچکترین خانه tbl نوع بیشاهدی را مشخص می‌کند.

یک عدد تصادفی با توزیع یکنواخت در بازه [a,b] تولید می‌کند.

```
int uniform(int a, int b)
  {
   int X;
   X = random(100);
   return (a+(b-a)*X/100)
  }

void init()
 {
   MW=20;
   MS=4;
   St=1000;
   tR=0;
   Qt=0;
   for(int i=0; i<20; i++)
    tbl[i] = uniform(100, 200).
    tbl[20] = St+1;
 }
void Kharabi()
 {
  if (MS>0)
    {
     MS = MS-1;
     tbl[code] = clock+ uniform(100,200);
    }
    else
     {
      MW= MW-1;
     tbl [code]= St+1;
     }
    if (tR==0)
     { tR=1;
(*)    tbl[20]=clock+ uniform (20,40);
      }
      else Qt++;
 }       cout << clock<<MW<<MS << tR<<Qt << "kharabi";
```

```
void payantamir()
{
  if (mw<20)
  {
   for( int k=0; k<20 ; k++)
    if (tbl[k] ==st+1)
      break;
   tbl[k]= cbck + uniform(100,200).
   mw= mw+1;
  }
  else
   ms=ms+1;
   if( Qt>0)
    {
     Qt--;
     tbl[code]= clock + uniform (20,40)
    }
    else
    {
     tR=0;   --تعداد تعمیرکار تغییر یابد-->  tR[code-20] = 0
     tbl[code]= st+1;
    }
   cout<< clock<<mw <<ms<< tR<< Qt<<"payan";
}
```

درصورتیکه در صورت سؤال بجای 20 ماشین مثال 30 ماشین گفته شود هم mw را در تابع مقداردهی اولیه برابر 30 قرار می دهیم و هم آرایه tbl دارای 31 خانه می شود و در تابع مقداردهی اولیه 30 خانه اول آن را مقداردهی می نمائیم.

• اگر تعداد تعمیرکارها بیش از یکی باشد اولاً بجای tR یک آرایه به اندازه تعداد تعمیرکارها در نظر می گیریم که وضعیت تعمیرکارها را در آن قرار دهیم و در تابع مقداردهی اولیه تمام تعمیرکارها را صفر یا آزاد می کنیم و در تابع خرابی بجای اینکه فقط tR را بررسی کنیم که آزاد است یا خیر باید هر تعداد تعمیرکار که داریم بررسی نمائیم. و کد آن بصورت صفحه بعد تغییر می یابد:

s.a.m

```
for (int i=0; i<3; i++)
    if (tR[i]==0)
    {
      tR[i]=1;
      tbL[20+i]=clock+uniform
      break;
    }
  if (i==3)
    Qt++;
```

بجای if در تابع Kharab; بگن (*)

در تابع پایان تعمیر بجای tR=0 ، tR[code-20]=0 قرار می دهم.

امتحان تا اینجا

جلسه ششم

↓ پایان ترم

نحوه تعیین گام زمانی یا مقدار ثابت را بیان کنید. (گروه بندی)

شبیه سازی به روش پردازش فرآیندها به زبان gpss :

زبان gpss اصولاً برای شبیه سازی سیستم های صف قرار داده شده است و دارای نسخه های متعددی است در زبان gpss برای شبیه سازی از تکنیک پردازش فرآیندها استفاده می شود در روش پردازش فرآیندها، فرآیندهایی که بر روی اشیاء سیستم از ابتدا تا انتها صورت می گیرد را شبیه سازی می کنیم.
در حالیکه در شبیه سازی به روش زمانبندی پیشامدها برنامه ها براساس پیشامدهایی که در سیستم اتفاق می افتاد تقسیم بندی می کردیم.

تعریف فرآیند :
به یک مجموعه پیشامدهای متوالی که به طریقی به هم مربوط باشند و یک نتیجه یا تولید کننده یک درآمد گفته می شود به طور مثال مجموعه پیشامدهایی که از لحظه ورود یک مشتری به سیستم تا خروج آن از سیستم صورت می گیرد یک فرآیند گفته می شود. در پردازش فرآیندها سیستم بصورت

مجموعه‌ای از فرآیندها در نظر گرفته می‌شود که به دلیل تفاوت در لحظات شروع آنها توالی، موازی و یا متداخل می‌باشد. در روش پردازش فرآیندها بجای اینکه شاهدها و تغییرات حاصل از پیشامدها را پردازش کنیم مراحل انجام فرآیندها را در نظر می‌گیریم. به عبارت دیگر یک مدل شبیه‌سازی به روش پردازش فرآیندها شامل دستوراتی برای انجام یک یا چند فرآیند می‌باشد که این دستورالعمل‌ها مجموعاً حرکت اشیاء موقت را در سیستم به منظور تکمیل یک فرآیند مشخص می‌سازند. هر دستورالعمل gpss یک فعالیت یا مرحله از فرآیند را شبیه‌سازی می‌کند.

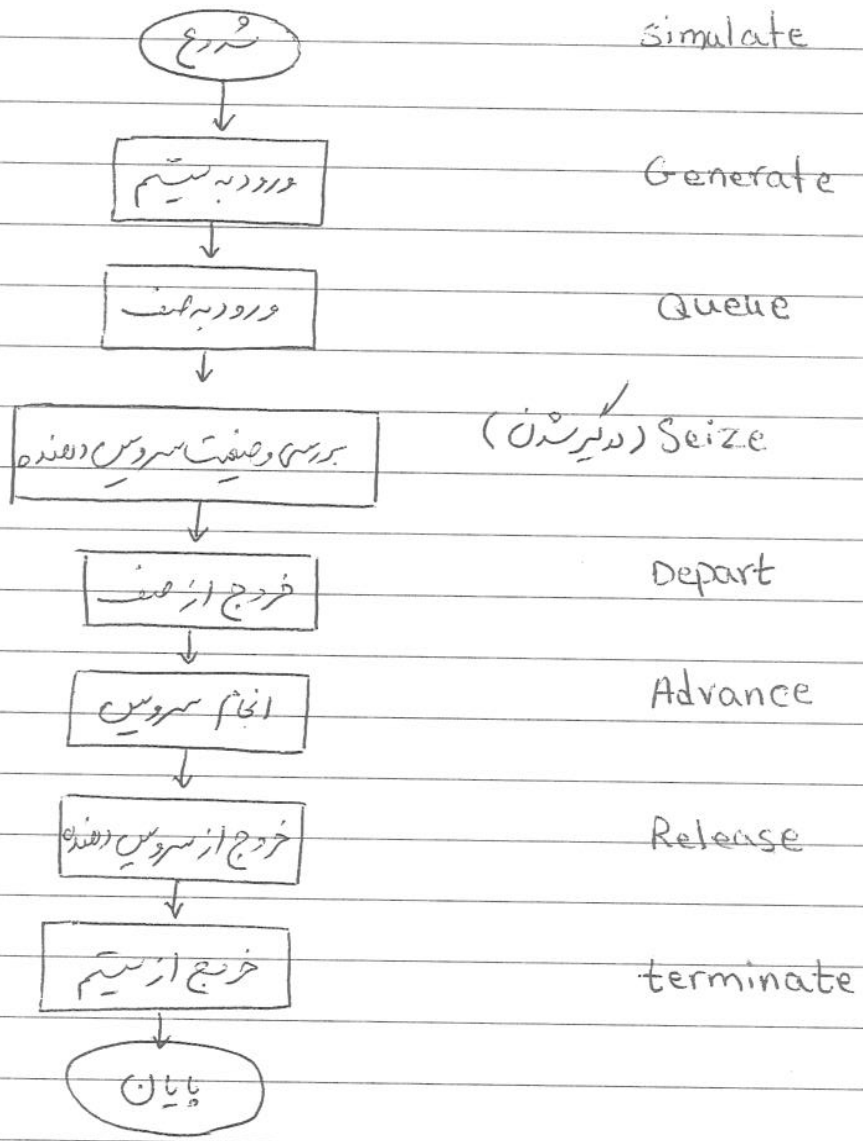
تعریف فعالیت:
به مجموعه عملیاتی که برای تبدیل وضعیت اشیاء در سیستم صورت گیرد یک فعالیت گفته می‌شود.
در زبان gpss اشیاء به دو دسته تقسیم می‌شوند:

1- اشیاء دائم یا سرویس دهنده:
- سرویس دهنده تکی یا وسیله (Facility)
- سرویس دهنده چندتایی یا انباره (storage)

2- اشیاء موقت transaction

قبل از اینکه به شرح دستورات زبان gpss بپردازیم برای آشنایی با ساختار کلی برنامه‌ی در زبان gpss و روش پردازش فرآیندها برنامه‌ی بسیار ساده بدون در نظر گرفتن جزئیات دستورات می‌نویسیم پس به شرح هریک از دستورات زبان gpss خواهیم پرداخت.

فلوچارت زیر مراحل لازم و ترتیب انجام مراحل آ ها در یک سیستم یک صف یک سرویس دهنده را نشان می‌دهد. این فلوچارت مسیری که یک شیء در سیستم طی می‌کند را نشان می‌دهد و نه ترتیب اجرای کامپیوتری برنامه‌ها. هریک از این مراحل توسط یک دستور از زبان gpss پیاده سازی می‌گردد که دستورات متناظر هریک از مراحل را در کنار آنها نوشتیم.



هریک از دستورات زبان gpss دارای پارامترهایی می باشد حال به شرح هریک از دستورات می پردازیم:

۱- دستور Generate :

برای ایجاد و ورود اشیاء به سیستم بطاری رود و شکل کلی آن بصورت زیر است :

Generate A, B, C, D, E, F, G, H, I

/ / /

پارامتر اول (A) : متوسط فاصله زمانی بین ورود اشیاء به سیستم را مشخص می کند. اگر فاصله زمانی بین ورود اشیاء دارای توزیع یکنواخت باشد، اگر فاصله زمانی بین ورود اشیاء از توزیع دیگری پیروی کند پارامتر A میانگین یا پارامتر آن تابع توزیع را نشان می دهد.

پارامتر دوم (B) : انحراف معیار یا اختلاف ابتدا و انتهای بازه نسبت به میانگین را مشخص می کند. اگر فاصله زمانی بین ورود دارای توزیع یکنواخت باشد. و اگر فاصله زمانی بین ورود دارای توزیع دیگری باشد بجای پارامتر B نام آن تابع توزیع را می نویسیم.
اگر پارامتر B را ننویسیم صفر در نظر گرفته می شود و اشیاء با فاصله زمانی دقیقاً برابر با پارامتر A وارد سیستم می شوند.

مثال)
Generate 20, 5
اشیاء با توزیع یکنواخت در بازه 15 تا 25 [15 25] وارد سیستم می شوند.

Generate 20
اشیاء 20 دقیقه به دقیقه وارد می شوند. یعنی با فاصله زمانی 20 وارد می شوند.
20, 40, 60, ...

Generate 10, FN$Expon
اشیاء با توزیع نمایی یا Expon با میانگین 10 وارد سیستم می شوند.

پارامتر C :
اگر زمان ورود اولین شی به سیستم با فاصله زمانی بین ورود سایر اشیاء متفاوت باشد پارامتر C را می نویسیم که زمان ورود اولین شی را مشخص می کند اگر پارامتر C را ننویسیم اولین شی نیز همانند سایر اشیاء وارد سیستم می شود.

پارامتر D :
حداکثر تعداد کل اشیاء که توسط این دستور ایجاد و وارد سیستم می شوند را نشان می دهد.
اگر این پارامتر را ننویسیم محدودیتی در تعداد اشیاء وجود نخواهد داشت.

پارامتر E :
اولویت یا حق تقدم اشیاء ورودی را مشخص می کند. که می تواند یک عدد صحیح بین 0 تا 127 باشد. اگر این پارامتر را ننویسیم بصورت پیش فرض صفر یا کمترین اولویت در نظر گرفته می شود.

پارامتر F تا I:
تعداد و نوع مشخصه های اشیاء را نشان می دهد. هر شیء می تواند ۴ نوع مشخصه داشته باشد:

pB	مشخصه یک باتی
pH	دوباتی (نیم‌گلدای)
pF	چهارباتی (تمام‌گلدای)
pL	اعشاری

مثال) عملکرد هریک از دستورات زیر را در زبان gpss بنویسید:

```
Generate  10,5,2,100,3,2pB,4pH,3pF,5pL
```

جواب) اشیاء با فاصله زمانی با توزیع یکنواخت در بازه [5,15] ایجاد و وارد سیستم می‌شود. تولین شیء دسته از ۲ واحد زمان وارد شده و کلاً ۱۰۰ شیء می‌توانند وارد سیستم شوند و اولویت کلیه اشیاء 3 می‌باشد و هر شیء دارای 2 مشخصه یک باتی، 4 مشخصه 2 باتی و 3 مشخصه ۴ باتی و 5 مشخصه اعشاری می‌باشد.

```
Generate  20,FN$Expon,5,,,4pB
```

اشیاء با فاصله زمانی با توزیع نمایی با میانگین 20 وارد سیستم می‌شوند. تولین شیء دسته از 5 واحد زمان وارد شده و محدودیتی در تعداد اشیاء ورودی وجود ندارد و اولویت کلیه اشیاء صفر می‌باشد و هر شیء دارای 4 مشخصه یک باتی است.

دستور Assign:

برای مقداردهی به مشخصه‌های اشیاء به کار می‌رود و شکل کلی آن بصورت زیر است:

```
Assign  A,B,C
```

پارامتر A: شماره مشخصه

پارامتر B: مقداری که می خواهیم در مشخصه قرار دهیم.

پارامتر C: نوع مشخصه را معین می کند.

مثال) در مشخصه اعشاری دوم شی ۲ مقدار 17.5 قرار می دهد ← Assign 2,17.5,pL

↓

یا pL[2] = 17.5

دستور priority:

برای تغییر مقدار تقدم اشیاء بکار می رود. و شکل کلی آن بصورت زیر است:

priority A

که A می تواند یک عددی بین ۰ تا 127 باشد. شی ای که به این دستور می رسد تقدم آن به مقدار A تغییر می یابد.

جلسه دهم

دستور Queue:

برای ورود اشیاء به صف استفاده می شود و شکل کلی آن به صورت زیر است:

Queue A,B

پارامتر A نام یا شماره صف می باشد. پارامتر B تعداد اشیایی است که وارد صف می شوند. در صورتیکه پارامتر B را ننویسیم یک در نظر گرفته می شود.

دستور Depart:

برای خروج اشیاء از صف بکاری رود و شکل کلی آن بصورت زیر است: Depart: A,B

پارامتر A نام یا شماره صف می باشد. پارامتر B تعداد اشیایی که از صف خارج می شوند.

دستور Seize:

برای بررسی وضعیت سرویس دهنده تکی یا وسیله بکار می رود. شی که به این دستور می رسد وضعیت

سرویس دهنده ها بررسی می کند
درصورتیکه آزاد باشد آن را اشغال می نماید. در غیر اینصورت در صف قبل از این دستور باقی می ماند

اگر تا سرویس دهنده آزاد گردد و شغل آن بصورت زیر است:

Seize نام سرویس دهنده کلی

دستور Release
برای خروج از سرویس دهنده کلی و آزاد نمودن آن بکار می رود و شغل کلی آن بصورت زیر است:

Release نام سرویس دهنده کلی

دستور Enter:
برای بررسی وضعیت سرویس دهنده چندتایی (انباره یا storage) و اشغال آن بکار می رود و شغل کلی آن بصورت زیر است:

Enter A,B

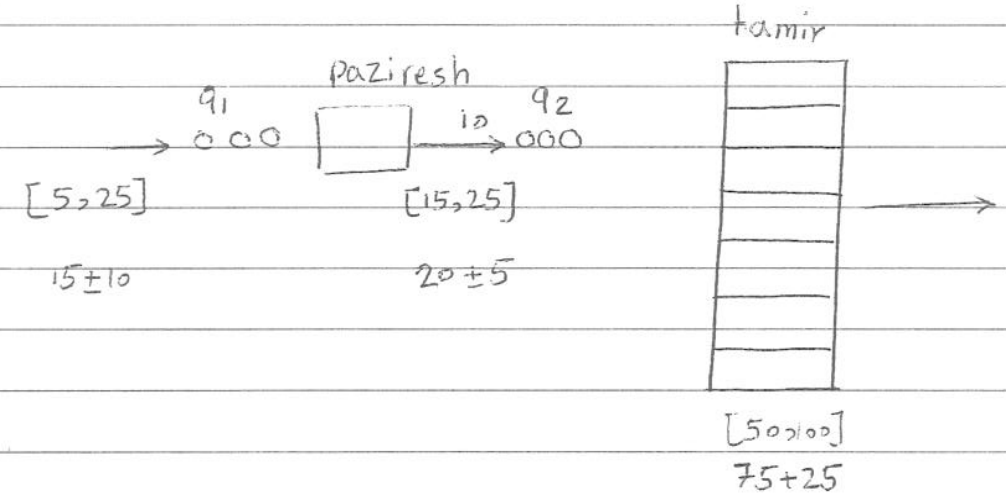
A نام انباره B: تعداد واحد مورد تقاضا از انباره می باشد.

همانگونه که گفته شد به این دستور می رسد و وضعیت انباره را بررسی می کند، درصورتیکه به تعداد واحد مورد تقاضا از واحدهای انباره خالی باشد آن را اشغال می نماید در غیر اینصورت در صف قبل از این دستور منتظر می ماند تا به تعداد کافی از واحدهای انباره آزاد گردد.

دستور Leave:
برای خروج و آزادکردن سرویس دهنده چندتایی یا انباره به کار می رود و شغل کلی آن به صورت زیر است:

Leave A,B

A نام انباره و B تعداد واحد از انباره که آزاد می گردد.



Simulate

```
tamir   storage   5

        Generate  15,10

        Queue     q1
        Seize     pazireh
        Depart    q1
        Advance   20,5
        Release   pazireh

        Advance   10

        Queue     q2
        Enter     tamir
        Depart    q2
        advance   75,25

        Leave     Tamir
        terminate 1
        start     50
```

سرویس دهنده تلفن

سرویس دهنده حیاط

متغیرها و توابع:

اگر بخواهیم در برنامه از متغیری استفاده نماییم نیازی به تعریف آن متغیر نمی‌باشد و در هر نقطه از برنامه که بخواهیم مقداری در آن متغیر قرار دهیم آن را تعریف می‌کنیم. نحوه تعریف آن بصورت زیر است:

نام یا شناسه متغیر variable مقدار یا عبارت که در متغیر قرار می‌دهیم.

مثال:

max	variable	0
A	variable	5*2
1	variable	20

اگر بخواهیم در عبارتهای بعدی از یک متغیر استفاده کنیم اگر نام متغیر یک رشته از حروف باشد قبل از نام متغیر حروف $V قرار می‌دهیم و اگر نام متغیر یک شناسه باشد قبل از نام متغیر حرف V را قرار می‌دهیم.

مثال بالا ↑

مثال) sum V1 + V$A (1 و A) Sum = 1+A = 30

اگر برای متغیرهای اعشاری بجای variable از fvariable استفاده می‌کنیم.

توابع:

در تورباپاسکال توابعی برای تولید اعداد تصادفی با توزیع یکنواخت وجود دارد ولی اگر نیازمند اعداد تصادفی با توزیع‌های دیگر در برنامه باشیم باید آن توابع توزیع را در ابتدای برنامه تعریف کنیم. نحوه تعریف توابع توزیع بصورت زیر است:

نام تابع function A,B

پارامتر A ورودی تابع و پارامتر B یکی از ترکیب‌های D_n و C_n می‌باشد که D برای توابع گسسته و C برای توابع پیوسته بکار می‌رود. و n تعداد زوج مرتب هایی را نشان می‌دهد که مقدار تابع را به ازای ورودیهای مختلف نشان می‌دهد.

مثال) تابعی بنویسید تا اعداد تصادفی با توزیع نمایی استاندارد ایجاد نماید.

Expon function Rn1, C12

0.0, 0.0 / 0.1, 0.104 / 0.2, 0.222 / 0.3, 0.355 / - - -

هرگاه تابع Expon را فراخوانی کنیم توسط تابع Rn1 که از توابع خود زبان می باشد یک عدد تصادفی با توزیع یکنواخت در بازه صفر و یک [0,1] تولید می کند و مقدار تابع را به آرگومان عدد از زوج مرتب های تابع بدست آورده و باز می گرداند.

برای فراخوانی یک تابع قبل از نام تابع $fn قرار می دهیم.

دستورات کنترلی:
در حالت معمول اشیاء موقت به همان ترتیب فیزیکی نوشته شدن دستورات برنامه حرکت می کند. دستوراتی که باعث تغییر مسیر اشیاء شده و کنترل اجرای برنامه را به نقطه دیگری از برنامه منتقل می کند دستورات کنترلی گفته می شود.

دستور transfer:

برای تغییر مسیر اشیاء بکار می رود و شکل کلی آن بصورت زیر است:
transfer A, B, C

A یک عبارت عددی بین صفر و یک می باشد. و B و C برچسب یا lable دو دستور از برنامه می باشد. شیئی که به این دستور می رسد به احتمال پارامتر A به دستور دارای برچسب C می رود و به احتمال 1-A به دستور دارای برچسب B می رود.
اگر پارامتر A را تنویسیم صفر در نظر گرفته می شود و به احتمال صد درصد به دستور دارای برچسب B می رود.

مثال) transfer 0.2 و l_1 و l_2

به احتمال 20٪ به l_2 و به احتمال 80٪ به l_1 می‌رود.

دستور test:

یکی دیگر از دستورات کنترلی است که برای تغییر مسیر اشیاء بکار می‌رود شکل کلی آن بصورت زیر است:

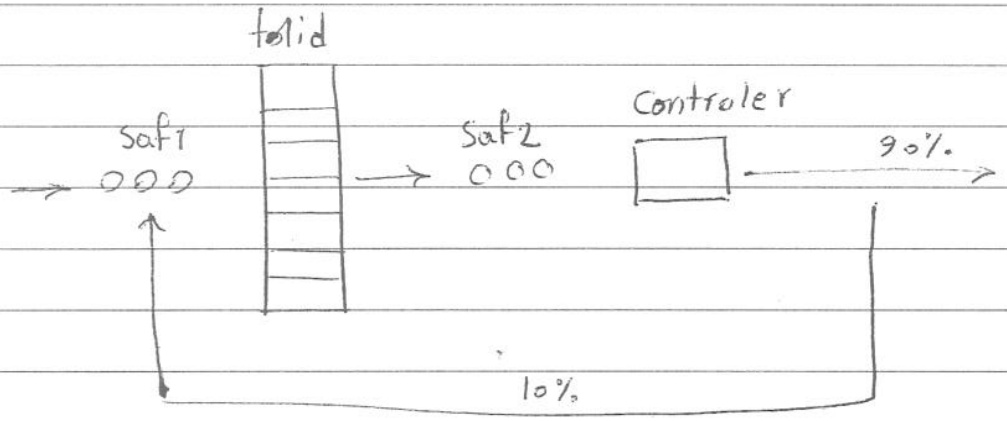
test X A,B,C

X می‌تواند یکی از شرایط زیر باشد:

E	مساوی
NE	نامساوی
L	کوچکتر
LE	کوچکتر مساوی
G	بزرگتر
GE	بزرگتر مساوی

A و B دو مشخصه عددی استانداردی باشد. و C برچسب یکی از دستورات برنامه است. هر شیئی که به این دستور می‌رسد مشخصه A و B با شرط X مقایسه می‌شود اگر شرط برقرار باشد به دستور بعدی می‌رود در غیر اینصورت به دستور دارای برچسب C می‌رود. اگر پارامتر C را ننویسیم شیئ در دستور قبل از این دستور منتظری ماند تا شرط برقرار گردد.

مثال) در یک کارگاه تولیدی 10 خط تولید وجود دارد. قطعات جهت تولید با توزیع نمایی با میانگین 20 وارد سیستم می‌شود. اگر هر یک از خطوط تولید آزاد باشند تولید آن محصول را شروع می‌کند در غیر اینصورت قطعات در یک صف منتظر می‌مانند تا یک خط تولید آزاد گردد. تولید هر محصول بین 100 تا 200 واحد زمان طول می‌کشد. محصولات تولیدی جهت کنترل نهایی به بخش کنترل می‌روند که در این بخش توسط یک کنترلر چک می‌شود. 90٪ محصولات سالم بوده و از سیستم خارج می‌شوند و 10٪ معیوب بوده و به خط تولید بازگردانده می‌شوند. محصولاتی که به خط تولید برمی‌گردند نسبت به قطعات اولیه تقدم بالاتری دارند. زمان کنترل هر محصول نیز به طور تصادفی با توزیع نمایی با میانگین 30 واحد زمان طول می‌کشد برنامه شبیه‌سازی این سیستم را تا خروج 500 محصول از سیستم به زبان gpss بنویسید.



تابع توزیع Expon function Rn1, C12

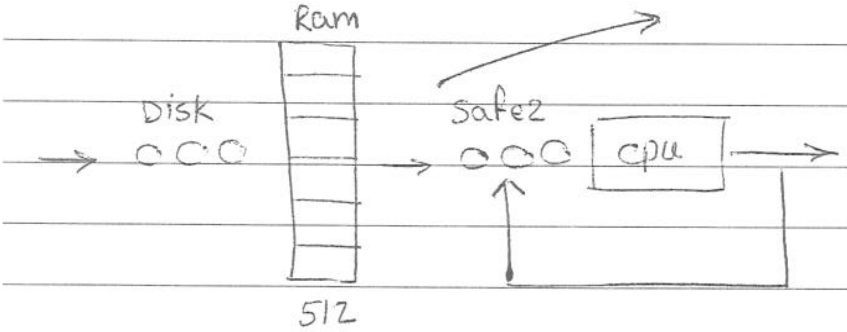
0.0, 0.0 / 0.1, 0.104 / 0.2, 0.222 / 0.3, 0.355 / ....

تعریف متغیر tolid Storage 10

	Generate	20, FN$ Expon
L1	Queue	Saf1
	Enter	tolid
	Depart	saf1
	Advance	150, 50
	leave	tolid
	Queue	saf2
	seize	Controler
	Depart	saf2
	Advance	30, FN$ Expon
	Release	Controler
	Transfer	0.9, L1, L2
L2	terminate	1
	start	500

مثال) در یک سیستم کامپیوتری 512 مگابایت Ram وجود دارد برنامه‌هایی که برای اجرا وارد این سیستم می‌شوند بین 10 تا 60 مگابایت فضا اشغال می‌کنند و زمان اجرای هریک از برنامه‌ها نیز بطور یکنواخت بین 20 تا 80 میلی ثانیه طول می‌کشد. برنامه‌ای در لحظه ورود درصورتیکه در حافظه به اندازه کافی فضا وجود داشته باشد در حافظه اجرا می‌شوند و منتظر اجرا توسط cpu می‌گردند در غیراینصورت بروی دیسک در یک صف منتظر می‌مانند تا به اندازه کافی حافظه آزاد گردد. cpu در برش‌های زمانی 2 میلی ثانیه‌ای به اجرای برنامه‌های داخل حافظه می‌پردازد و پس از هر دور که 2 میلی ثانیه هر برنامه را اجرا می‌کند دور بعد را شروع می‌کند این دورها آنقدر ادامه می‌یابد تا به ترتیب برنامه‌ای خاتمه یافته و از حافظه نیز خارج گردد.

برنامه‌هایی که زمان اجرای آنها کمتر از 30 میلی ثانیه است نسبت به سایر برنامه‌ها از تقدم بالاتری برخوردارند.



مثال) برنامه شبیه‌سازی این سیستم را برای 6000 میلی ثانیه به زبان gpss بنویسید.
فاصله زمانی بین ورود برنامه‌ها به سیستم دارای توزیع غیر یکنواخت با میانگین 25 میلی ثانیه است.

```
var1    variable    20+60*Rn1       var1    عدد تصادفی بین 20 تا 80
var2    variable    10+50*Rn2       var2    عدد تصادفی بین 10 تا 60
Expon   function    RN3,C12
0.0,0.0/0.1,0.104/0.2,0.222/0.3,0.355/---

Ram     storage     512
        simulate
        generate    25,FN$Expon,,,2pf
        assign      1,V$var1,pF
*       test  L     pF1,30,Li
        priority    2
L1      assign      2,V$var2,pF
```

	Queue	Disk
	Enter	Ram , pF2
	Depart	Disk
L_2	Queue	Saf 2
	Sieze	cpu
	Depart	Saf 2
	test G	pF1 , 2 , L3
	Advance	2
	Release	cpu
	Assign	1- و 2 و pF →
	Transfer	و L2 →
L_3	Advance	pF1
	Release	cpu
	leave	Ram , pF2
	terminate	0
	Generate	6000
	terminate	1
	start	1

به علت اینکه شرط اجرای پایان برنامه رسیدن به 6000 واحد زمان است نه خروج تعداد برنامه بنابراین در لحظه 6000 یک شی ء Generate می کنیم و بلافاصله این شی ء را Terminate می کنیم و start صفر شده و برنامه خاتمه می یابد.

اگر شرط پایان برنامه خروج 500 برنامه (اجرای) از رسیدن به لحظه 6000 باشد بجای 4 دستور انتهای برنامه چنین می نویسیم:

terminate	1
Generate	6000
terminate	500
start	500

در این حالت هر بار که یک برنامه خارج می شود یک واحد از پارامتر start کم می شود تا به صفر برسد. اگر در لحظه 6000 هنوز پارامتر start صفر نشده باشد به بیکاره 500 واحد از آن

terminate (کم می کند) و برنامه خاتمه می یابد.

اگر برنامه های دارای زمان اجرای کمتر از 30 واحد بیشترین تقدم و برنامه های بین 30 تا 50 واحد زمان اجرا دارای تقدم کمتر و برنامه های بالاتر از 50 واحد زمان دارای کمترین تقدم باشند بایستی دو دستور * می نویسیم:

test L pF1, 50, L1
priority 1 → زمان اجرا < 50 در غیر اینصورت به 4 برو
L test L pF1, 30, L1
priority 2

else