PROJETO 2019

O projeto a ser implementado durante o ano letivo é um sistema de informação geográfica simplificado, como definido abaixo.

> *A geographic information system (GIS) is a computer system for capturing, storing, checking, and displaying data related to positions on Earth's surface. GIS can show many different kinds of data on one map. This enables people to more easily see, analyze, and understand patterns and relationships.*
> **Fonte: http://www.nationalgeographic.org/encyclopedia/geographic-information-system-gis/**

O sistema será implementado incrementalmente e em fases. É importante enfatizar que em cada fase o sistema evolui, isto é, novas funcionalidades são acrescentadas, requisitos de implementação podem ser mudados, porém, as funcionalidades existentes **devem continuar funcionais**.

LEMBRETE: AVALIAÇÃO (PROGRAMA DO CURSO)

A avaliação será feita por meio de provas (P) e de trabalhos individuais(Ti) e em equipe (Te). Serão atribuidas notas numéricas às avaliações (0-10). A avaliação poderá ser feita considerando apenas o artefato produzido (texto, programa, etc), como também considerando sua apresentação oral. Espera-se que no caso de trabalhos em equipe, todos os membros trabalhem assiduamente na execução do trabalho. Em caso de dúvidas sobre a efetiva participação de um membro, o professor poderá argui-lo durante a apresentação do trabalho ou convocar a equipe para uma entrevista. Caso o aluno não demonstre efetivo domínio do trabalho produzido, sua nota poderá ser diferente da dos outros membros da equipe. Muita atenção, neste caso, sua nota provavelmente será muito baixa (não excluída a nota nula).

O peso das atividades individuais (provas e trabalhos) será 7. O peso das atividades em equipe será 4. A última atividade do ano letivo terá peso maior, visando diminuir a possibilidade de o aluno ser aprovado sem entregar/realizar a última atividade do ano.
Caso a última atividade seja individual, o peso será 11; caso seja em equipe, o peso será 6.

O que é esperado do aluno:

1. frequência em todas as aulas e pontualidade. Isto é muito importante, pois as aulas são planejadas como uma sequência de atividades que incrementalmente farão os alunos adquirirem os conhecimentos e habilidades desejadas.
2. Participação efetiva nos trabalhos. Isto também é muito importante, pois muitas habilidades e conteúdos só serão adquiridos pela prática. A forma de organização da equipe deve possibilitar que todos os membros da equipe tenham o mesmo nível de aprendizado.
3. Estudo sério da parte teórica. Esta é uma disciplina de Ciência da Computação em uma Universidade. Espera-se que o aluno dedique algumas horas semanais para o aprofundamento da teoria exposta. Espera-se que o aluno estude a parte teórica antes de que inicie a implementação da parte prática.
4. Que o aluno não use apenas as transparências para estudar. O professor disponibilizará as transparências usadas apenas como forma de orientar os estudos. As transparências, propositalmente, contém apenas o esqueleto do conteúdo e figuras para facilitar a explicação do professor. Espera-se que o aluno faz uso efetivo dos livros listados na bibliografia.

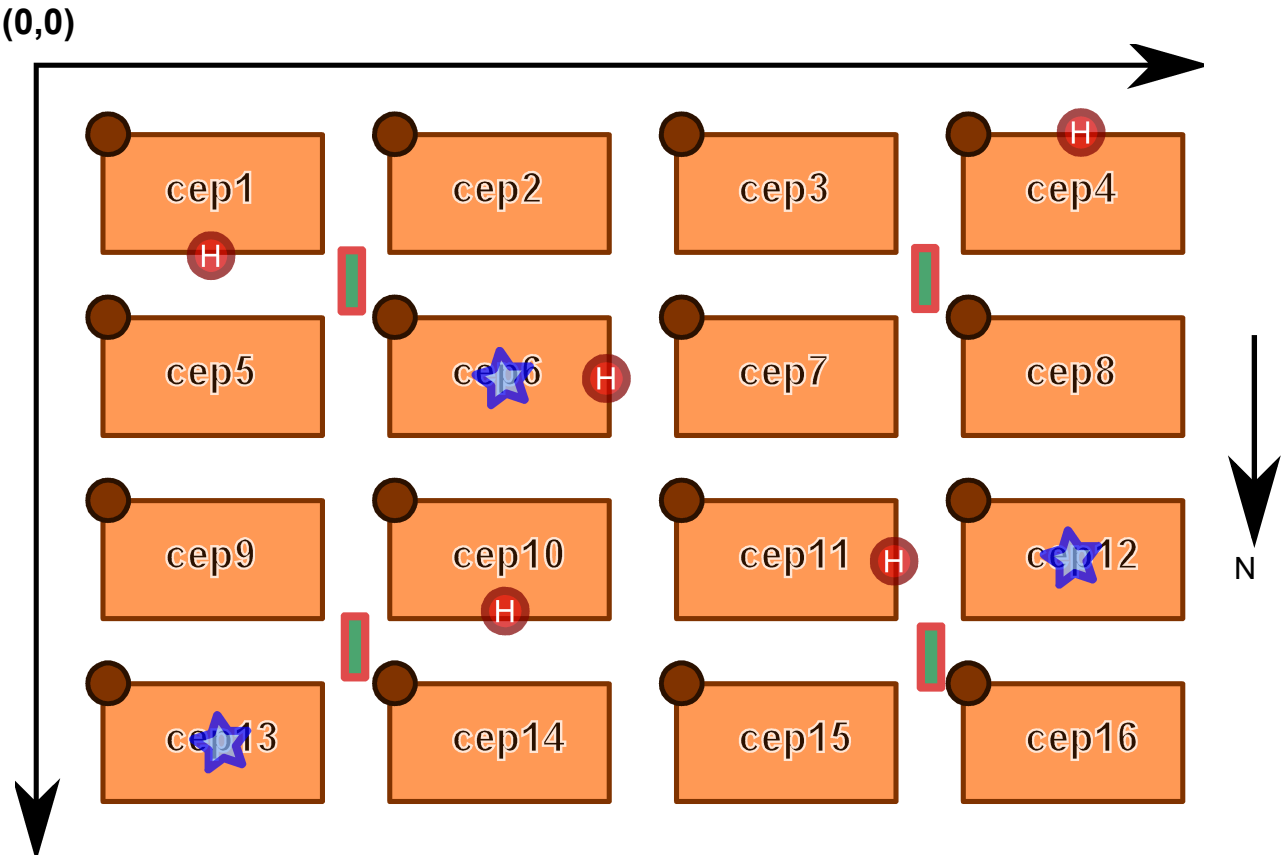
IMPORTANTE: não será tolerado nenhum tipo de fraude nas provas ou nos trabalhos. Qualquer fraude detectada durante ou após sua a realização será punida com nota zero a todos os envolvidos.

DESCRIÇÃO

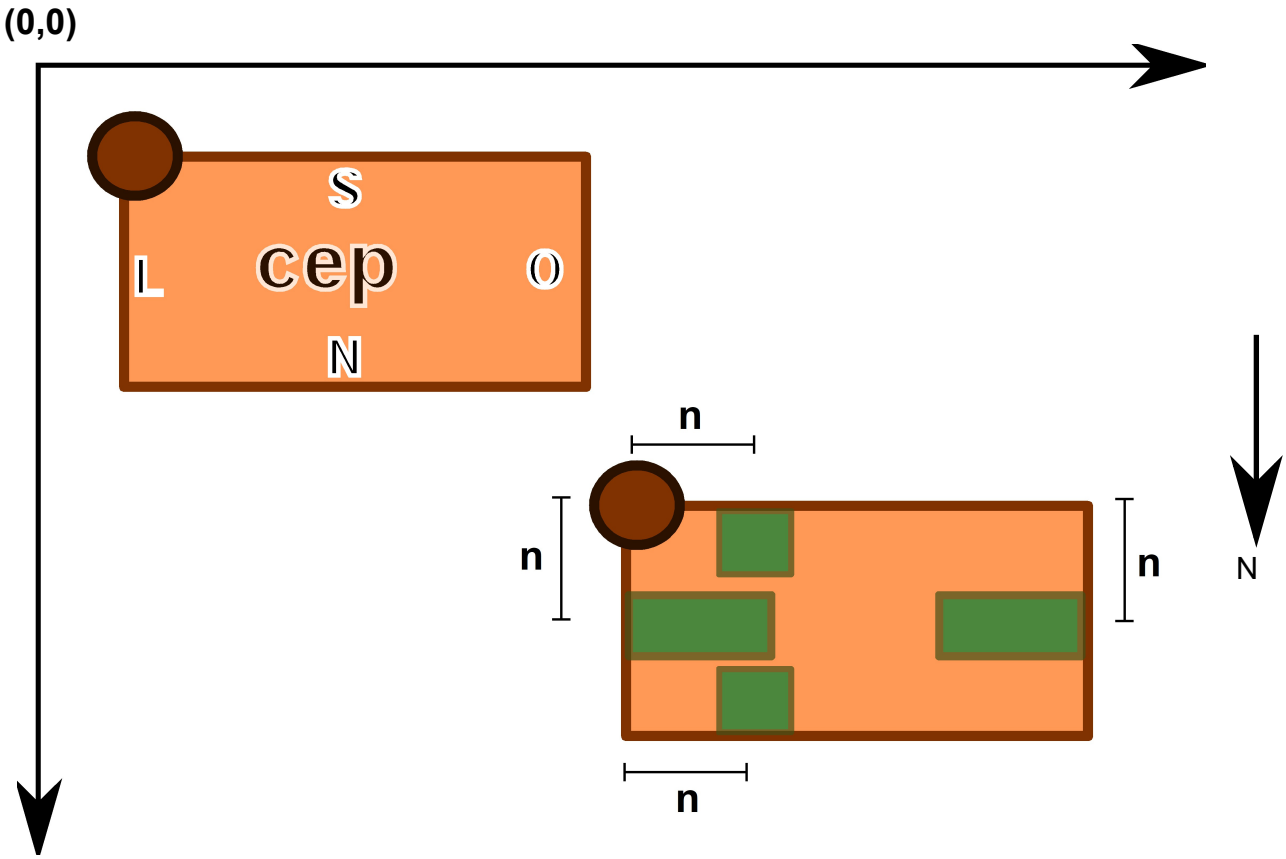
Um Sistema de Informações Geográficas (SIG), para a nossa finalidade, é um sistema que contém (não exclusivamente) dados geo-referenciados, isto é, dados com algum atributo de localização espacial (uma coordenada).

O sistema manipulará o mapa de uma cidade e algumas informações relacionadas.

O mapa de uma cidade é composto por um conjunto de retângulos que representam as quadras; e, um conjunto de equipamentos urbanos (hidrantes, semáforos, torres de celular, pontos de ônibus, etc). Cada equipamento urbano é localizado no mapa por um único ponto, conforme o exemplo abaixo.



A cidade exemplificada acima chama-se **Bitnópolis** e possui 16 quadras. O sistema de endereçamento de Bitnópolis é inspirado no de nossa capital federal. Cada **quadra** possui 4 **faces** (N,S,L,O) e é identificada por um **CEP** alfanumérico. O número de uma casa ou estabelecimento comercial é a **distância** da frente da casa até uma projeção do ponto de ancoragem do retângulo que representa a quadra (veja figura abaixo). Assim, um endereço é da forma CEP/Face/número, por exemplo, cep15/S/45. O ponto de ancoragem do retângulo é o canto sudeste da quadra.



ENTRADA DE DADOS

A entrada de dados, via de regra, ocorrerá por meio de um ou mais arquivos. Estes arquivos estarão sob um diretório, referenciado por **BED** neste texto.[1]

SAIDA DE DADOS

O dados produzidos serão mostrados na saída padrão e/ou em diversos arquivos-texto. Alguns resultados serão gráficos no formato SVG. Os arquivos de saída serão colocados sob um diretório, referenciado por **BSD** neste texto.[2]

ORGANIZAÇÃO DA ENTREGA

O trabalho deve ser submetido no formato **ZIP**, cujo nome deve ser curto, mas suficiente para identificar o aluno ou a equipe.[3] Este arquivo deve estar organizado como descrito à frente.

PROCESSO DE COMPILAÇÃO E TESTES DO TRABALHO

Organização do ZIP a ser entregue

A organização do zip a ser entregue pelo aluno deve ser a seguinte:

1 Indicado pela opção -e.
2 Indicado pela opção -o.
3 Por exemplo, jrsilva.zip (se aluno se chamar José Roberto da Silva), jrsilva-mrcarneiro.zip (para uma equipe com dois alunos. Evite usar maiúsculas, caracteres acentuados ou especiais.

[abreviatura-nome]	*Por exemplo, jrsilva.*
LEIA-ME.txt	*colocar matrícula e o nome do aluno. Atenção: O número da matricula de estar no início da primeira linha do arquivo. Só colocar os números; não colocar qualquer pontuação.*
*	*Outros arquivos podem ser solicitados a cada fase.*
/src	*(arquivos-fonte)*
makefile	*deve ter target para a geração do arquivo objeto de cada módulo e o target* **siguel** *que produzirá o executável de mesmo nome dentro do mesmo diretório* **src**. *Os fontes devem ser compilados com a opção -fstack-protector-all.* ** adotamos o padrão C99. Usar a opção -std=c99.*
*.h e *.c	***Atenção:*** *não devem existir outros arquivos além dos arquivos fontes e do makefile*

Organização do diretório para a compilação e correção dos trabalhos (no computador do professor):

[HOME_DIR]	
*.py	*scripts para compilar e executar*
\t	*diretório contendo os arquivos de testes*
*.geo *.qry	*arquivos de consultas, talvez, distribuídos em alguns outros sub-diretórios*
\alunos	*(contém um diretório para cada aluno)*
\abrnome	*diretório pela expansão do arquivo submetido (p.e., jrsilva)*
	outros subdiretórios para os arquivos de saída informados na opção -o

Os passos para correção serão os seguintes:

1. O arquivo .zip será descomprimido dentro do diretório alunos, conforme mostrado acima
2. O makefile provido pelo aluno será usado para compilar os módulos e produzir o executável. Os fontes serão compilados com o compilador gcc em um máquina virtual Linux. Os executáveis devem ser produzidos no mesmo diretório dos arquivos fontes O professor usará o GNU Make. Serão executadas (a partir dos scripts) o seguinte comando:
   - **make siguel**
3. O programa será executado automaticamente várias vezes: uma vez para cada arquivo de testes e o resultado produzido será inspecionado visualmente pelo professor. Cada execução produzirá (pelo menos) um arquivo **.svg** diferente dentro do diretório informado na opção **-o**. Possivelmente serão produzidos outros arquivos .svg e .txt.

APENDICE

https://www.gnu.org/software/make/manual/make.html
http://opensourceforu.com/2012/06/gnu-make-in-detail-for-beginners/

FASE I

A Entrada

A entrada do algoritmo será basicamente um conjunto de retângulos e círculos dispostos numa região do plano cartesiano e, possivelmente, algumas consultas, por exemplo, que indagam se duas das formas geométricas se sobrepõem. Os comandos estão contidos num arquivo .geo e as consultas num arquivo .qry.

Considere a Ilustração 1. Cada forma geométrica é definida por uma coordenada âncora e por suas dimensões. A coordenada âncora do círculo é o seu centro e sua dimensão é definida por seu raio. A coordenada âncora do retângulo é seu canto inferior esquerdo[4] e suas dimensões são sua largura e sua altura. As coordenadas que posicionam as formas geométricas são valores reais e estão contidas dentro da região delimitada pelos cantos (0,0) e (x_{max},y_{max}). Cada forma geométrica é indentificada por um número inteiro.

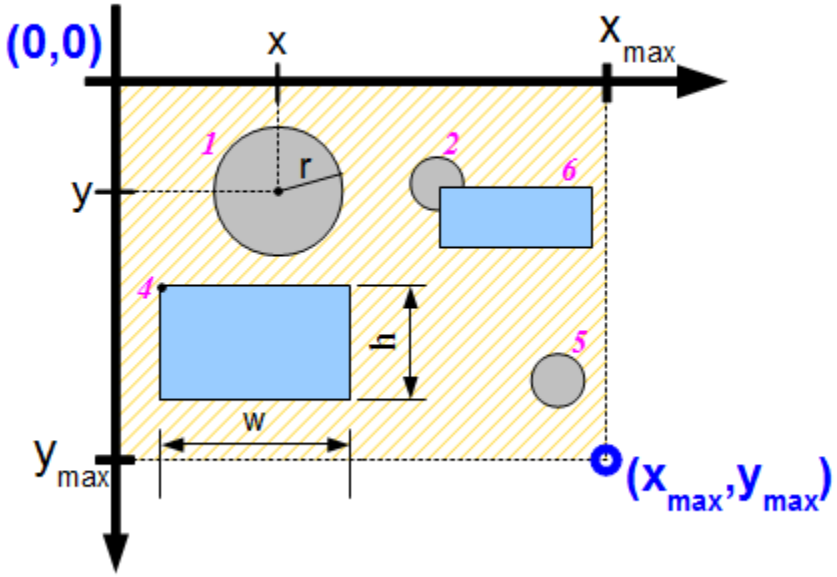


Ilustração 1

A tabelas abaixo mostram os formato dos arquivos de entrada (.geo e .qry). Os arquivos de entrada são compostos, basicamente, por conjunto de comandos (um por linha), a saber: **`c`** (desenhe um círculo), **`r`** (desenhe um retâgulo), **`t`** (escreva um texto), etc.

Cada comando tem um certo número de parâmetros. Os parâmetros mais comuns são:

- `i,j,k`: número inteiro, maior ou igual a 1. Identificador de uma forma geométrica criada pelos comandos **`c`** ou **`r`**.
- `r`: número real. Raio do círculo.
- `x,y`: números reais. Coordenada (x,y).
- `cor`: string. Cor válida dentro do padrão SVG.[5]

4 Note que o plano cartesiano está desenhado "de ponta-cabeça" em relação à representação usual.

5 http://www.december.com/html/spec/colorsvg.html.
https://www.w3.org/Graphics/SVG/IG/resources/svgprimer.html

comando	parâmetros	descrição
nx	`i`	*Altera o número máximo de círculos e retângulos (i.e., c + r) criados no arquivo. O valor default é 1000.*
c	`i r x y cor1 cor2`	*desenhar círculo. cor1 é a cor da borda e cor2 é a cor do preenchimento*
r	`i w h x y cor1 cor2`	*desenhar retângulo: w é a largura do retângulo e h, a altura. cor1 é a cor da borda e cor2, a do preenchimento*
t	`x y texto`	*todo o texto até o final*
comandos .geo		

comando	parâmetros	descrição
o?	`j k`	*As formas geométricas cujos indentificadores são* **`j`** *e* **`k`** *se sobrepõem?*[6] ***Saida*: *.txt:* *copiar o comando e, na linha seguinte escrever uma mensagem informando se sobrepõe ou não*** ***.svg*: *traçar um retângulo que envolva ambas figuras, de bordas tracejadas, se não se sobrepõem; linha cheia, se se sobrepõe***
i?	`j x y`	*O ponto (x,y) é interno à j-ésima forma geométrica?*[7] ***Saída*: *.txt*: *copiar o texto da consulta e, na linha seguinte, informar se é interno ou não.*** ***.svg*: *colocar um ponto (pequeno círculo) nas coordenadas (x,y) e pintá-lo de de verde se for interno e vermelho se for externo. Colocar uma linha ligando o ponto ao centro de massa da figura j.***
d?	`j k`	*Qual é a distância entre os centro de massa das formas geométricas* **`i`** *e* **`j`***?* ***Saída*: *.txt*: *copiar o texto da consulta e, na linha seguinte, informar a distância.*** ***.svg*: *traçar uma reta entre os centros de massa das duas figuras e escrever próximo da metade da linha a distância***

6 A borda da figura pertence à figura. Assim, as figuras que coincidem apenas nas bordas também se sobrepõem.
7 Um ponto na borda da figura pertence à figura, **mas não é interno** à figura.

comando	parâmetros	descrição
bb	sufixo cor	*Cria um novo arquivo svg contendo os círculos e retângulos referentes aos comandos c e r processados até o momento.* ***Para cada círculo****, deve ser criado um retângulo que o envolve ("bounding box").* ***Para cada retângulo*** *deve ser criada uma elipse envolvida por tal retangulo. As figuras produzidas deve ser preenchidas com a cor* **cor**. *O nome do arquivo gerado deve ser* **nomebase-*sufixo.svg***.
comandos .qry		

A figura abaixo mostra um exemplo de um arquivo de entrada (consistente com a Ilustração 1). Note que a extensão do arquivo é **.geo**. As primeiras operações desenham círculos e retângulos. Na parte parte final do arquivo, estão colocadas duas consultas de sobreposição. A primeira pergunta se o círculo 2 e o retângulo 6 se sobrepõem (de fato, sim) e, a segunda, pergunta se os círculos 1 e 5 se sobrepõem (de fato, não). O último comando solicita que seja produzido um outro arquivo **.svg** (a01-lnhs2.svg) mostrando linhas originárias do centro de massa da figura 2 até cada uma das outras figuras, anotando na respectiva linha o seu comprimento.

```
c 1 50.00 50.0 30.00 grey magenta
r 6 121.0 46.0 100.0 30.0 cyan yellow
c 2 grey magenta 120.0 45.0 15.0
r 4 10.0 150.0 90.0 40.0 cyan yellow
c 5 230.0 180.0 13.0 grey magenta
```

***a01*.geo**

```
o? 2 6
o? 1 5
d? 2 4
i? 5 210.0 160.0
bb suf01 green
```

q.qry

A Saída

O programa deverá produzir um arquivo **.svg** e um arquivo **.txt** ambos com o mesmo nome base do arquivo **.geo**.

O arquivo .svg produzido deve mostrar as formas geométricas. Além disso, para o comando **o**, caso as figuras identificadas se sobreponham, elas devem ser envolvidas por um retâgulo tracejado contendo a palavra "sobrepoe". Existem várias ferramentas que renderizam arquivos .svg. As figuras abaixo mostram um exemplo de arquivo **.svg** e sua respectiva renderização.

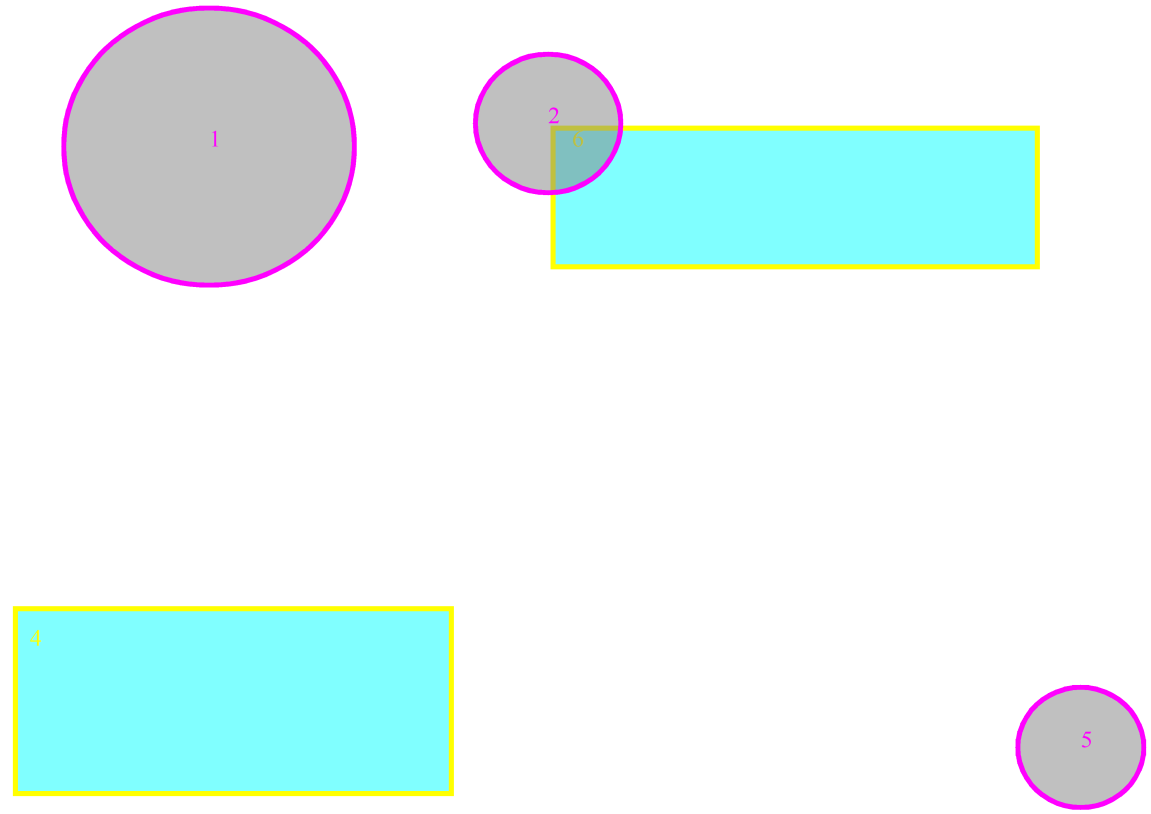
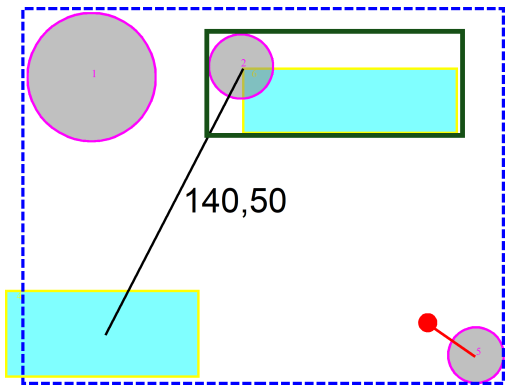
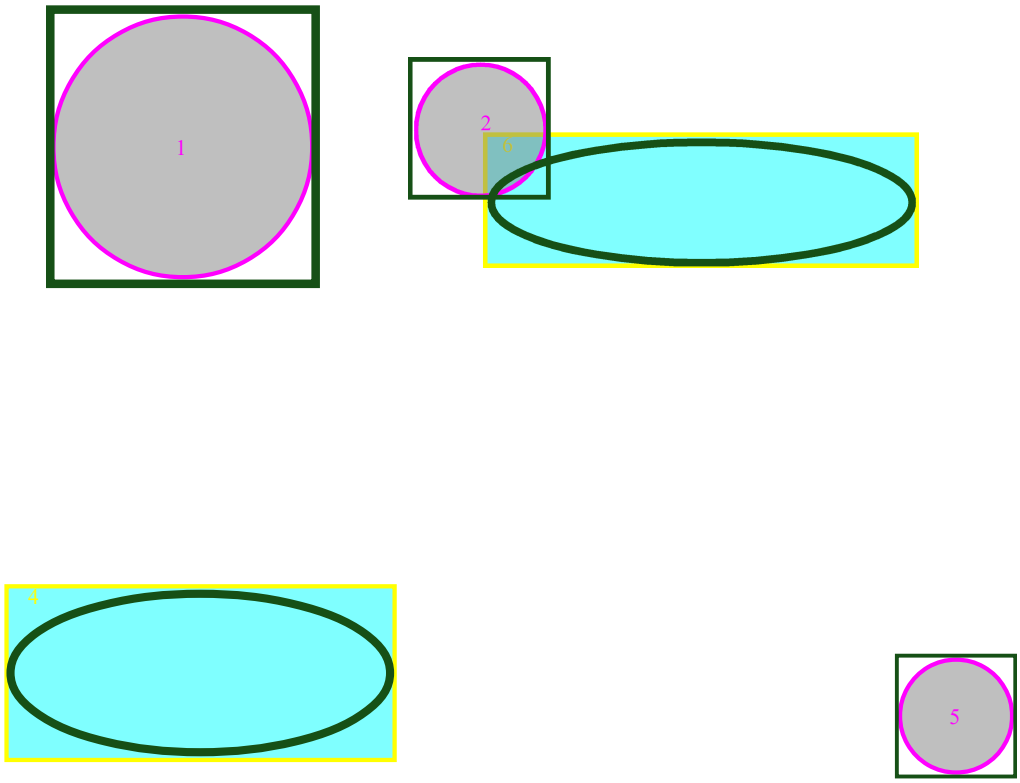


Illustration 2: Arquivo a01.svg

Se o arquivo .geo (a01.geo, no nosso exemplo) for processado em conjunto com um arquivo .qry (q.qry, no exemplo), além de a01.svg, deverá ser produzido o arquivo a01-q.svg, contendo os círculo, retângulos, etc acrescidos aos resultados da consulta. Como no exemplo:



Caso o arquivo .qry contiver o comando bb, deverá ser produzido um terceiro arquivo .svg (a01-q-suf01.svg), contendo os círculo, retângulos, etc, acrescido do resultado do comando bb, como no exemplo:



O processamento de um arquivo `.geo` deve também produzir um arquivo-texto (a01-q.txt, no exemplo) contendo o resultado textual de todas as consultas. Neste arquivo deve ser copiado em uma linha o texto da consulta e, na linha seguinte, o seu resultado.

```
o? 2 6
SIM

o? 1 5
NAO

d? 2 4
140,50

i? 5 210.0 160.0
NAO INTERNO
```

Arquivo a01-q.txt

FASE II

Nesta fase serão acrescentados quadras e equipamento urbanos. A nossa cidade começa a tomar forma. Temos quadras, hidrantes, radio-bases de telefonia móvel e semáforos.

Nesta fase serão acrescentados novos comandos ao arquivo **`.geo`**:

comando	parâmetros	
q	`cep x y w h`	*Insere uma quadra (retângulo e cep)*
h	`id x y`	*Insere um hidrante*
s	`id x y`	*Insere um semáforo*
rb	`id x y`	*Insere uma rádio-base (torre de celular)*
cq	`cfill cstrk sw`	*Cores do preenchimento (`cfill`) e da borda (`cstrk`) das quadras, espessura da borda (`sw`) (a partir deste comando)*
ch	`cfill cstrk sw`	*Cores do preenchimento e da borda (e sua espessura) dos hidrantes (a partir deste comando)*
cr	`cfill cstrk sw`	*Cores do preenchimento e da borda (e sua espessura) das torres de celular (a partir deste comando)*
cs	`cfill cstrk sw`	*Cores do preenchimento e da borda (e sua espessura) dos semáforos (a partir deste comando)*
sw	`cw rw`	*Espessuras das bordas, rexpectivamente, dos círculos (comando c) e dos retângulos (comando r) (a partir deste comando)*
nx	`i nq nh ns nr`	*Informa o número máximo de formas(círculos e retângulos), quadras, hidrantes, semáforos e rádio-bases, respectivamente, criados no arquivo. O valor default é 1000.*
Novos comandos do arquivo .geo		

Alguns comandos de atualização e consulta podem ser colocados em um arquivo `.qry`:

comando	parâmetros	
dq	`(L1 | L2) id r`	*remove todas quadras que estiverem inteiramente dentro a uma distância de no máximo r do equipamento urbano identificado por id segundo a métrica* `L1` *ou* `L2`*. determinado pelos parâmetros do comando.* ***No arquivo .svg:*** *as quadras removidas não devem aparecer. O equipamento urbano me questão deve enfatizado com um anel grosso de duas cores.* ***No arquivo .txt****, deve apresentar os ceps das quadras removidas, o id e respectivas informações do equipamento urbano.*
del	`( cep | id )`	*Remove a quadra, hidrante, semáforo ou torre de identificação* `id` *(ou* `cep`*).* ***No arquivo .svg****: quadra ou equipamento urbano removido não deve aparecer.* ***No arquivo .txt****: reportar os dados relacionados da quadra ou equipamento urbano removido.*
cbq	`x y r cstrk`	*Muda a cor da borda para* `cstrk` *de todas as quadras que estiverem inteiramente contidas dentro do círculo de centro em (x,y) e de raio* `r`*.* ***No arquivo .svg:*** *quadras eleitas pintadas conforme descrito.* *Reporta no* ***arquivo .txt*** *o cep das quadras que tiveram a cor da borda alterada*
crd?	`( cep | id )`	*Imprime no* ***arquivo .txt*** *as coordenadas e a espécie do equipamento urbano de um determinado cep ou com uma determinada identificação.*
trns	`x y w h dx dy`	*Desloca todas as quadras e equipamentos urbanos que estiverem inteiramente dentro do retângulo (x,y,w,h) em dx unidades no eixo x e dy unidades no eixo y. Note que dx e dy podem ser medidas negativas.* *No* ***arquivo .svg****: quadras e equipamentos urbanos devem aparecer nas novas posições.* *No* ***arquivo .txt:*** *cep e id das quadras e equipamentos movidos, bem como as respectivas posições anteriores e atualizadas.*
Comandos do arquivo .qry		

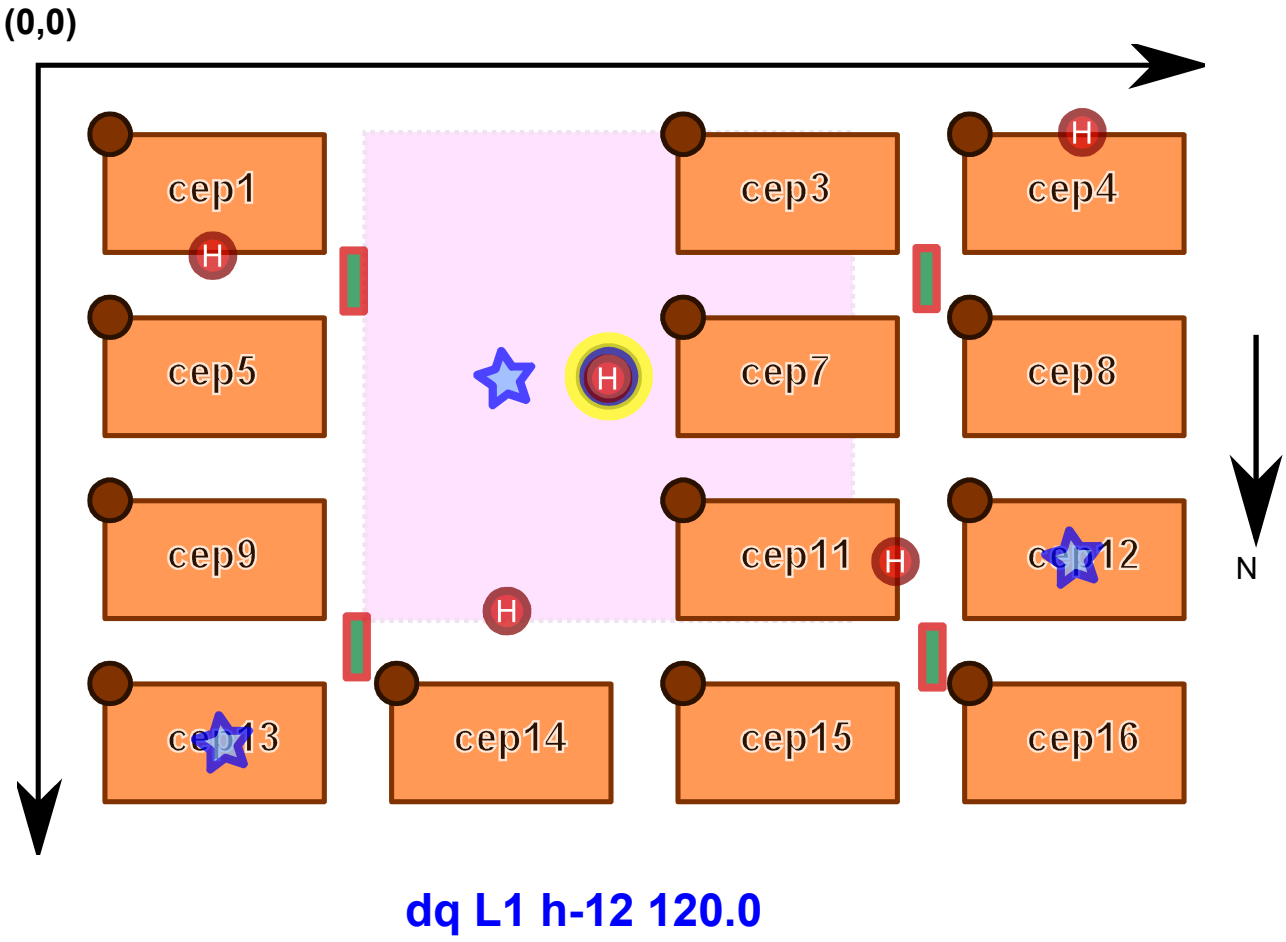
EXEMPLOS

```
cq blue black 2px
ch red yellow 1px
ct black red 3px
q cep_001-10 37.00 15.00 89.00 40.00
q cep_001-20 137.00 15.00 89.00 40.00
q cep_001-30 237.00 15.00 89.00 40.00
cq yellow green 1.5px
q cep_002-10 37.00 115.00 89.00 40.00
q cep_002-20 137.00 115.00 89.00 40.00
q cep_002-30 237.00 115.00 89.00 40.00
h h-12 320.00 60.00
c 3 29.00 720.00 458.00 green yellow
r 4 5.00 29.00 1049.00 479.00 green blue
sw 3.4px 1.3px
r 5 55.00 42.00 215.00 702.00 green red
c 6 51.00 139.00 635.00 chocolate black
```

a1.geo

```
dq L1 h-12 120.0
crd? cep_001-10
crd? h-12
```

q1.qry



dq L1 h-12 120.0

A SAIDA

A arquivo `.geo` (com os novos comandos) deve continuar produzindo as mesmas saídas da fase anterior (com as modificações aprensentadas nesta fase). As quadras devem ser renderizados como retângulos e pintadas como determinado pelo comando `cq`. Os outros equipamentos urbanos devem ser similarmente pintados com as cores de contorno e de preenchimento informadas pelos respectivos comandos.

A IMPLEMENTAÇÃO

O TAD Listas mostrado em sala de aula deve ser completamente implementado. As implementação da lista **deve** ser a estática duplamente encadeada. As quadras e equipamentos urbanos devem ser armazenados em listas diferentes (ou seja, são 4 listas diferentes).

Outros TADs **devem** ser projetados.

Todos os TADs **devem** ser implementados como módulos, conforme descrito em sala de aula (.h, .c)

É **expressamente proibida** a declaração de structs em arquivo .h.

O Programa

O nome do programa deve ser **siguel** e aceitar quatro parâmetros:

siguel [-e *path*] -f *arq*.geo [-q consulta.qry] -o *dir*

O primeiro parâmetro (**-e**) indica o diretório base de entrada. É opcional. Caso não seja informado, o diretório de entrada é o diretório corrente da aplicação. O segundo parâmetro (**-f**) especifica o nome do arquivo de entrada que deve ser encontrado sob o diretório informado pelo primeiro parâmetro . O terceiro parâmetro (-q) é um arquivo de consultas. O último parâmetro (**-o**) indica o diretório onde os arquivos de saída (***.svg e *.txt**) deve ser colocados. Note que o nome do arquivo pode ser precedido por um caminho relativo; ***dir e path*** é um caminho absoluto ou relativo (ao diretório corrente).

A seguir, alguns exemplos de possíveis invocações de **siguel**:

- `siguel -e /home/ed/testes/ -f t001.geo -o /home/ed/alunos/aluno1/o/`
- `siguel -e /home/ed -f ts/t001.geo -o /home/ed/alunos/al1/o`
- `siguel -f ./tsts/t001.geo -e /home/ed -o /home/ed/alunos/aluno1/o/`
- `siguel -o ./alunos/aluno1/o -f ./testes/t001.geo`
- `siguel -o ./alunos/aluno1/o -f ./testes/t001.geo -q ./t001/q1.qry`
- `siguel -e ./testes -f t001.geo -o ./alunos/aluno1/o/ -q ./q1.qry`

A Avaliação

Espera-se uma atitude pró-ativa para a aquisição dos conhecimentos (i.e., estudo) para resolver o problema proposto.

A avaliação se baseará em dois critérios: **(a)** inspeção do código-fonte (especial atenção na modularização); **(b)** compilação e testes do executável.

O Que Entregar

Submeter a sala Moodle o arquivo .zip com os fontes , conforme descrito anteriormente.

RESUMO DOS PARÂMETROS DO PROGRAMA SIGUEL

Parâmetro / argumento	Opcional	Descrição
`-e` ***`path`***	S	Diretório-base de entrada (**BED**)
`-f` ***`arq`*****`.geo`**	N	Arquivo com a descrição da cidade. Este arquivo deve estar sob o diretório **BED**.
`-o` ***`path`***	N	Diretório-base de saída (**BSD**)
`-q` ***`arqcons`*****`.qry`**	S	Arquivo com consultas. Este arquivo deve estar sob o diretório **BED**.

RESUMO DOS ARQUIVOS PRODUZIDOS

-f	-q	comando com sufixo	arquivos
arq.geo			arq.svg
arq.geo	***arqcons***.qry		arq.svg arq-arqcons.svg arq-arqcons.txt
arq.geo	***arqcons***.qry	***sufx***	arq.svg arq-arqcons.svg arq-arqcons.txt arq-arqcons-sufx.[svg|txt][8]

ATENÇÃO:
* o fontes devem ser compilados com a opção `-fstack-protector-all`.
* adotamos o padrão C99. Usar a opção `-std=c99`.

8 Podem ser produzidos o respectivo arquivos .svg e/ou .txt, dependendo da especificação do comando.