Elementos de Programação - IST - LMAC, MEBiom

20 de Dezembro de 2019 Ficha 3A Duração: 15m

Número: _____ **Nome:** _____

Considere *actividades* e *calendários*, com as seguintes operações:

- `activ(ti,tf)`: actividade que dura do instante `ti` ao instante `tf` (`ti`<`tf`);
- `inic(a)`: instante inicial da actividade `a`;
- `fim(a)`: instante final da actividade `a`;
- `novo()`: calendário vazio;
- `adiciona(a,c)`: calendário que resulta de adicionar a actividade `a` ao calendário `c`;
- `numactivs(c)`: número de actividades calendarizadas em `c`;
- `prim(c)`: actividade calendarizada em `c` (não vazio) com menor instante inicial (uma delas, caso haja várias);
- `elimprim(c)`: calendário que resulta de eliminar do calendário (não vazio) `c` a actividade `prim(c)`.

a) Em *Python*, pretende-se representar cada actividade como um par `(ti,tf)`, e cada calendário como uma lista de actividades ordenada por ordem crescente do seu instante inicial. Apresente implementações eficientes apenas para as operações `adiciona`, `prim` e `elimprim`.

(vsff)

b) Desenvolva, sobre a camada de abstracção obtida acima e assegurando a independência da implementação, uma função `tempolivre` que recebendo um calendário `c` e instantes `ti,tf` (`ti`<`tf`), devolve o tempo livre total disponível (sem actividades calendarizadas em `c`) entre `ti` e `tf`.

Elementos de Programação - IST - LMAC, MEBiom

20 de Dezembro de 2019 Ficha 3B Duração: 15m

Número: _____ **Nome:** _____

Considere *actividades* e *calendários*, com as seguintes operações:

- `activ(ti,tf)`: actividade que dura do instante `ti` ao instante `tf` (`ti`<`tf`);
- `inic(a)`: instante inicial da actividade `a`;
- `fim(a)`: instante final da actividade `a`;
- `novo()`: calendário vazio;
- `calendariza(a,c)`: calendário que resulta de adicionar a actividade `a` ao calendário `c`;
- `vazioQ(c)`: `True` se e só se o calendário `c` está vazio;
- `prim(c)`: actividade calendarizada em `c` (não vazio) com menor instante inicial (uma delas, caso haja várias);
- `elimprim(c)`: calendário que resulta de eliminar do calendário (não vazio) `c` a actividade `prim(c)`.

a) Em *Python*, pretende-se representar cada actividade como um par `(ti,tf)`, e cada calendário como uma lista de actividades ordenada por ordem de calendarização. Apresente implementações eficientes apenas para as operações `calendariza`, `prim` e `elimprim`.

(vsff)

b) Desenvolva, sobre a camada de abstracção obtida acima e assegurando a independência da implementação, uma função `dmedia` que recebendo um calendário (não vazio) `c`, devolve a duração média das actividades calendarizadas em `c`.

Elementos de Programação - IST - LMAC, MEBiom

20 de Dezembro de 2019 Ficha 3C Duração: 15m

Número: _____ **Nome:** _____

Considere *actividades* e *calendários*, com as seguintes operações:

- `activ(ti,tf)`: actividade que dura do instante `ti` ao instante `tf` (`ti`<`tf`);
- `inic(a)`: instante inicial da actividade `a`;
- `fim(a)`: instante final da actividade `a`;
- `novo()`: calendário vazio;
- `adiciona(a,c)`: calendário que resulta de adicionar a actividade `a` ao calendário `c`;
- `numactivs(c)`: número de actividades calendarizadas em `c`;
- `prim(c)`: actividade do calendário (não vazio) `c` com menor instante inicial (uma delas, caso haja várias);
- `elimprim(c)`: calendário que resulta de eliminar do calendário (não vazio) `c` a actividade `prim(c)`.

a) Em *Python*, pretende-se representar cada actividade como um par `(ti,tf)`, e cada calendário como uma lista de actividades ordenada por ordem crescente do seu instante final. Apresente implementações eficientes apenas para as operações `adiciona`, `prim` e `elimprim`.

(vsff)

b) Desenvolva, sobre a camada de abstracção obtida acima e assegurando a independência da implementação, uma função `liberta` que recebendo um calendário `c` e instantes `ti,tf` (`ti`<`tf`), devolve o calendário que resulta de eliminar de `c` todas as actividades que tenham sobreposição com o intervalo entre `ti` e `tf`.

Elementos de Programação - IST - LMAC, MEBiom

20 de Dezembro de 2019 Ficha 3D Duração: 15m

Número: _____ **Nome:** _____

Considere *actividades* e *calendários*, com as seguintes operações:

- `activ(ti,tf)`: actividade que dura do instante `ti` ao instante `tf` (`ti`<`tf`);
- `inic(a)`: instante inicial da actividade `a`;
- `fim(a)`: instante final da actividade `a`;
- `novo()`: calendário vazio;
- `calendariza(a,c)`: calendário que resulta de adicionar a actividade `a` ao calendário `c`;
- `vazioQ(c)`: `True` se e só se o calendário `c` está vazio;
- `prim(c)`: actividade do calendário (não vazio) `c` com menor instante inicial (uma delas, caso haja várias);
- `elimprim(c)`: calendário que resulta de eliminar do calendário (não vazio) `c` a actividade `prim(c)`.

a) Em *Python*, pretende-se representar cada actividade como um par `(ti,tf)`, e cada calendário como uma lista de actividades ordenada por ordem inversa de calendarização. Apresente implementações eficientes apenas para as operações `calendariza`, `prim` e `elimprim`.

(vsff)

b) Desenvolva, sobre a camada de abstracção obtida acima e assegurando a independência da implementação, uma função `dmax` que recebendo um calendário (não vazio) `c`, devolve a duração máxima de alguma actividade calendarizada em `c`.