

The tikzquests package

A parametric questions' repository

Version 2.3

Alceu Frigeri*

November 2025

Abstract

This is a framework for building parametric questions' repositories, which can be further used to construct parametric questions for exams. Unlike other packages (for instance *exam*, *exam-n* or *exam-lite*) this doesn't try to enforce any pre-defined presentation format, focusing only on how to set a repository and use it.

Contents

1	Introduction	1
2	Package Options	2
3	Repositories	2
4	Questions	3
4.1	Defining a Question	3
4.2	Using a Question	4
5	Parameters as keys and auxiliary macros.	4
5.1	Assigning a value to pre-defined keys	5
5.2	Assigning a value to new keys	5
5.3	Using a parameter key	5
6	Examples of Use	5
6.1	Package Options	5
6.2	A More Complete Example	6
6.3	Listing all Questions	9

1 Introduction

One recurring problem when typesetting exams, specially in Engineering fields, is the need to have parameterized questions, specially parameterized schematics used in exam's questions. The intent of this is to allow the creation of such repositories (one or more) with 'easily' parameterized text and/or graphic (*tikz*) questions, without enforcing any style/format whatsoever, leaving it to the end user.

The package offers

- A set of commands to create and use repositories,
- A set of commands to create and use 'questions' (code snippets) in a repository,
- Question's customization, parametrization, based on *keys* and a set of associated macros.

*<https://github.com/alceu-frigeri/tikzquests>

2 Package Options

<code>draft</code>	(default: false). When using a question (see 4.2), the question’s associated remarks and annotations will be printed as well, if any.
<code>in review</code>	This is an alias to <code>draft</code>
<code>local defs</code>	(default: false). If set, the pre-defined keys will be defined locally (at each command use, those at 4.2), instead of globally (default). This is less prone to name crashing, but comes with a significant performance hit.
<code>no defs</code>	(default: false). This changes the parametrization strategy suppressing the creation of “keys commands”.
<code>old settings</code>	(default: false). This changes the assignment behaviour when <code>no defs</code> is set. See 5
<code>keys prefix</code>	This will add a prefix to all variables. It defaults to ‘nothing’ (no prefix).
<code>keys</code>	(default: R,L,C,Z,Y,G,H,X,K,T,Q,D,EQ) This (re)sets the list of pre-defined <i>keys</i> . See 5.
<code>source keys</code>	(default: V,I) This (re)sets the list of pre-defined source <i>keys</i> (those with a <i>f</i>). See 5.
<code>xtrakeys</code>	This allows to expand the set of pre-defined (<i>keys</i>). See 5.
<code>idx</code>	(default: {},a,b) This (re)sets the list of pre-defined indexes per key. The {} corresponds to the single letter idx’s (a to z). See 5.
<code>xtraidx</code>	This allows to expand the set of pre-defined (<i>keys</i>) indexes per key. See 5.
<code>no alias</code>	(default: false). This suppress the creation of Question’s aliases (see 4.1), handy when one wants to reduce the ‘noise’ while listing all currently known Questions.
<code>undef color</code>	(default: red). This sets the color used to indicated the use of a non defined parameter. See 5.
<code>keys info</code>	(default: false). If set, when setting the pre-defined keys, the effective sets of <i>keys</i> and <i>indexes</i> are logged as a package info.

Warning: When (re)setting keys/idx list, don’t forget to insert the list between braces, otherwise the keys/indexes will be interpreted as package options leading to error messages when loading the package.

3 Repositories

Questions are stored in repositories. There is a pre-defined one, named `default`. Each repository has two sub-sets (a) a non-starred sub-set and (b) a starred sub-set. The rational behind it is to be able to separate the text part (e.g. the question enunciate) from the graphics part. It is suggested (but not enforced) to store the text part in the starred sub-set and the graphics part (*tikz*) in the non-starred sub-set. A repository can have an “unlimited” set of questions.

Note: All commands defined in 4.2 can use both subsets, except `\QuestionsList` which assumes the non-starred sub-set has only text code, whilst the starred sub-set has *tikz* commands.

<code>\defRepository</code>	<code>\defRepository* {<new-repository>}</code>
<code>\SelectRepository</code>	<code>\SelectRepository {<repository>}</code>

updated: 2025/10/01

`\defRepository` creates a new Repository, `<new-repository>`. The starred version also switches to it, making it the “current default”. `\SelectRepository` activates `<repository>` as the “current default” one.

It is possible to construct a tree of related repositories by using a slash, “/”, like “repoA”, “repoA/subA”, “repoA/subB”, “repoA/subC”, which can be handy when listing the questions of a repository, see 4.2.

Note: About repository’s names: It can be almost anything, the name can contain strings normally not allowed in a macro name, like spaces, dots, two-dots and so on. an important exception the backslash is still an active character, meaning that if someone typesets `\XYZ` as a repository, the value stored in `\XYZ` will be used (if `\XYZ` isn’t defined this might raise a rather cryptic L^AT_EX 2_ε error).

Note: Do not create a repository with an ending slash, like “repoName/”. Given the way sub-repositories are defined, it will render impossible to list this repository questions, see 4.2

Note: A warning is raised if the (deprecated) `\defNewRepository` is called. Use `\defRepository` instead.

Note: When creating a new repository, a warning is raised in case `<new-repository>` already exists. When Selecting a repository, an error is raised if `<repository>` doesn't exist.

4 Questions

As said, for each repository there are two sub-sets of questions (code snippets): starred ones, though for text (which may include L^AT_EX 2_ε commands) and non starred ones though for graphics (for instance, to be used inside a `tikz` environment).

Since the code re-factoring (version 2.0), the commands `\defQuestion`, `\defQuestionAlias`, `\Question`, `\tikzQuestion` and `\ftikzQuestion` use/refer to the “non-starred” sub-set, while the commands `\defQuestion*`, `\defQuestionAlias*`, `\Question*`, `\tikzQuestion*` and `\ftikzQuestion*` use/refer to the “starred” sub-set.

All questions can be parameterized (see 5 below) with a set of predefined keys.

4.1 Defining a Question

<code>\defQuestion</code>	<code>\defQuestion [<repository>] {<quest-name>} {<code>} [<remarks>]</code>
	<code>\defQuestion* [<repository>] {<quest-name>} {<code>} [<remarks>]</code>

`\defQuestion` creates a new Question in `<repository>` (per default using the “current active” one). `<quest-name>` will be the associated question key, for later reference. `<code>` will be the corresponding Question's code. `<remarks>` is just a small note associated with the Question. It won't be normally used/typeset, except if the `in review` option is being used, or when using the command `\QuestionsList`.

The starred version `\defQuestion*` will create the question in the “starred sub-set”, whilst the non starred one `\defQuestion` will create the question in the “non starred sub-set”.

Note: Since the code re-factoring (version 2.0) all commands in 4.2 can use both sets of questions, with one exception: `\QuestionsList` which assumes that the starred set stores text and the non-starred stores `tikz` graphics.

Note: About question's names: It can be almost anything, the name can contain strings normally not allowed in a macro name, like spaces, dots, two-dots and so on, including backslashes, meaning that if someone typesets `\XYZ` as a question name, `\XYZ` will be it's name: a backslash isn't an active character anymore and one can't use macros when defining a question's name.

Note: An error is raised if `<repository>` doesn't exist. In case the `<quest-name>` already exists, it's code is silently replaced by the new one.

<code>\defQuestionAlias</code>	<code>\defQuestionAlias [<dst-repository>] {<quest-alias>} [<org-repository>] {<quest-name>}</code>
updated: 2025/04/25	<code>\defQuestionAlias* [<dst-repository>] {<quest-alias>} [<org-repository>] {<quest-name>}</code>

`\defQuestionAlias` creates an alias, `<quest-alias>` (at `<dst-repository>`), for a given question, `<quest-name>` (from `<org-repository>`). The current active repository is used if none is specified. If only `<dst-repository>` is specified, the same repository is used for `<org-repository>`. As with `\defQuestion`, the “star” refers the sub-set being used.

Note: Be aware that this will copy the question definition at the alias creation point.

Note: An error is raised if `<dst-repository>`, `<org-repository>` or `<quest-name>` doesn't exist.

4.2 Using a Question

<code>\Question</code>	<code>\Question [⟨repository⟩] {⟨quest-name⟩} [⟨key=value list⟩] <⟨annotation⟩></code>
<code>\tikzQuestion</code>	<code>\tikzQuestion (⟨scale⟩) [⟨repository⟩] {⟨quest-name⟩} [⟨key=value list⟩] <⟨annotation⟩></code>
<code>\ftikzQuestion</code>	<code>\ftikzQuestion (⟨scale⟩) [⟨repository⟩] {⟨quest-name⟩} [⟨key=value list⟩] <⟨annotation⟩></code>
<code>updated: 2025/10/01</code>	<code>\Question* [⟨repository⟩] {⟨quest-name⟩} [⟨key=value list⟩] <⟨annotation⟩></code>
	<code>\tikzQuestion* (⟨scale⟩) [⟨repository⟩] {⟨quest-name⟩} [⟨key=value list⟩] <⟨annotation⟩></code>
	<code>\ftikzQuestion* (⟨scale⟩) [⟨repository⟩] {⟨quest-name⟩} [⟨key=value list⟩] <⟨annotation⟩></code>

Those are the main commands to display a question. The star/non-star defines which sub-set will be used. `\Question` is the “raw” base variant, the question’s code will be used, as is, in a local group. The starred version `\Question*` is meant for text questions, whilst the non-starred version `\Question` is meant for the case one wants to use a different graphic engine (for instance SVG, instead of the `tikz` as in `\tikzQuestion`). Note that this is just a suggestion, and not enforced.

The `\tikzQuestion` will display the question’s code inside a `tikzpicture` environment. And then, `\ftikzQuestion` will further nest the code inside a `center` environment (preparing it to be used inside a floating environment).

The `<annotation>` will only be added if, and only if, the `in review` option is being used. The `<scale>` factor, when present, is related to the current `\textwidth`, so a factor of 0.25 will scale the width of the question to 1/4th of the text width. The `<key=value list>` is a set of `keys`, see 5 below.

Note: Besides the key/parameters from 5 there is a special key `tikz keys` which can be used to pass `tikz` specific parameters for the underlying environment, this key is ignored when using `\Question` or `\Question*`.

Warning: A Warning is raised if one uses the old (deprecated) `\rawtikzQuestion` or `\textQuestion`. Use `\Question` instead.

Note: An error is raised if `<repository>` or `<quest-name>` doesn’t exist.

<code>\QuestionsFmtList</code>	<code>\QuestionsFmtList [⟨repositories list⟩] {⟨star-cmd⟩} {⟨nonstar-cmd⟩}</code>
<code>\QuestionsList</code>	<code>\QuestionsList [⟨repositories list⟩]</code>

`new: 2025/10/03`

`<repositories list>` is a comma separated list of repositories. For each repository, all Questions will be typeset in a `’describe’` alike environment. If no `<repositories list>` is supplied, all repositories will be listed.

`<star-cmd>` will be used to typeset a “star” question, it can be any command that accepts an optional `<repository>` name and a mandatory `<quest-name>`, like `\starcmd [⟨repository⟩] {⟨quest-name⟩}`.

Likewise `<nonstar-cmd>` will be used to typeset a “non star” question (assuming the same syntax).

`\QuestionsList[repo-list]` is a shortcut for

`\QuestionsFmtList[repo-list]{\Question*}{\tikzQuestion(0.35)}`

Note: If a `<repositories list>`’s term finishes with a slash, “/”, it will be treated as a family of repositories and all sub-repositories will be listed too.

Note: An error is raised if any repository in `<repositories list>` doesn’t exist. To reduce the clutter (and number of questions listed) one should consider the use of the `no alias` option.

5 Parameters as keys and auxiliary macros.

By default the set of `keys` and macros is defined as follow: `<prefix>⟨ID⟩⟨idx⟩`. Default `<prefix>` being nothing. The predefined range of `<IDs>` being *R, L, C, Z, Y, G, H, X, K, T, Q, D, EQ, V* and *I*. The predefined range of `<idx>` spans from a, b, c up to z, aa, ab, ac up to az and, finally, ba, bb, bc up to bz.

Note: That means, one gets to use `keys` as, for example, *Ra*=200, *Lca*=500, *Kbe*=230 and so on. For each of the key’s `<ID>` there is (per default) a set of 3x26 keys. If one adds some 3 other `<idx>` then one gets 6x26 keys per `<ID>`.

Note: The `<ID>`s can be extended with the `xtrakeys` option. For example with `xtrakeys={NN,B}`, each extra `<ID>` will add 3x26 keys (per default).

Note: The `<idx>`s can be extended with the `xtraidx` option. For instance with `xtraidx={f,g}` one gets *fa*, *fb*, *fc* up to *fz*, *ga*, *gb*, *gc* up to *gz* as well, a x26 set per extra `idx`.

Unless the `no defs` option is used, a corresponding macro (with the same name) will also be defined. For example, there is a macro `\Ra` associated with the key *Ra*. All those macros/keys are

initialized as follow (math mode) `\ensuremath{\langle prefix \rangle ID_{\langle idx \rangle}}`, resulting in $prefixID_{idx}$. So, for example, the key `Rab`, which can be accessed with the macro `\Rab`, will be predefined as R_{ab} , `Lca` (`\Lca`) will be predefined as L_{ca} , and so on.

Note: If `keys prefix` (see 2) is set, all *pre-defined* keys will have that prefix, but only them. For instance, if one set's `keys prefix = kp` (when loading this package), the same example above would be, key `kpRab`, macro `\kpRab`, default value kpR_{ab} , and so on.

The idea is that, when using the commands in 4.2, if one doesn't specify a `\langle key=value list \rangle`, the default values will be en force, and all one has to do (to change those values) is to set said list, which doesn't have to be complete, non assigned keys will keep their default value.

Besides those default keys, one can set and use any key at will (in `\langle key=value list \rangle`) non-existing keys will be created “on the fly” with the given name as the default value.

5.1 Assigning a value to pre-defined keys

In fact, with each and every predefined key there are 3 ways to assign a value to it `\langle key \rangle = value`, `\langle key* \rangle = value` and `\langle key raw \rangle = value`. The difference being that `\langle key raw \rangle` will assign whatever code/value to the key (and associated macro), `\langle key* \rangle` will assign the code/value inside a math environment.

warning: `\langle key \rangle = value` (without any specifier) will be the same as `\langle key* \rangle` (default). But, if the options `no defs` and `old settings` are used, it will be equivalent to `\langle key raw \rangle`.

5.2 Assigning a value to new keys

When using `\langle key \rangle = value`, if the `\langle key \rangle` isn't one of the pre-defined ones, then `value` will be assigned “as is” to `\langle key \rangle`.

warning: Note that, in this case there are no `\langle key* \rangle` or `\langle key raw \rangle` equivalents.

5.3 Using a parameter key

When defining the `\langle code \rangle` of a question (see 4.1) one has two options to recover a key's value:

- A macro named after the key itself (in case of the default keys, see above) and, or
- the `\QuestVal` command which allows to recover the value of both the default keys, as well as the ones defined on the fly.

Warning: If the option `no defs` is used, the only option to recover a key's value is `\QuestVal`. Users are advised to choose a style, and keep it (when starting new docs).

`\QuestVal` ★ `\QuestVal {\langle prefixIDidx \rangle}`
`\QuestVal {\langle key \rangle}`

This will always recover the value of a key, regardless if the key is one of the pre-defined ones (in the form `\langle IDidx \rangle`) or an “on the fly” new one, `\langle key \rangle`. If the key didn't get (re)defined with the `\langle key=value list \rangle`, it will return the key/parameter default value. In the case of a “on the fly” key, it will be the key's name in red (or the color set up with the `undef color` option, see 2).

Note: If `keys prefix` (see 2) is set, all *pre-defined* keys will have that prefix, it's part of their name.

Note: `\QuestVal` is fully expandable, if the key's code also is.

6 Examples of Use

6.1 Package Options

Package Options

```
\usepackage{tikzquests}
```

This is the default case, in which both `\QuestVal` and (for default keys) associated macro name can be used to retrieve a key/parameter value, and no prefix will be added to any key.

Package Options

```
\usepackage[xtrakeys={EX,N},xtraidx={f,h},undef color={blue},no defs]{tikzquests}
```

In this case, one will get the following set of keys

- Ra, Rb ... Rz, Raa, Rab ... Raz *Rfa, Rfb ... Rfz, Rha, Rhb ... Rhz*
- all other default sets of keys, plus
- EXa, EXb ... EXz, EXaa, EXab ... EXaz *EXfa, EXfb ... EXfz, EXha, EXhb ... EXhz*
- Na, Nb ... Nz, Nab, Naa ... Naz *Nfa, Nfb ... Nfz, Nha, Nhb ... Nhz*

Besides that, the undefined color will be blue and no additional macro will be defined, meaning that `\QuestVal` is the only option to retrieve the value of a key/parameter.

Package Options

```
\usepackage[xtrakeys={EX,N},keys prefix = {pr}]{tikzquests}
```

In this case, one will get the following set of keys

- prRa, prRb ... prRz, prRaa, prRab ... prRaz
- all other default sets of keys, plus
- prEXa, prEXb ... prEXz, prEXaa, prEXab ... prEXaz
- prNa, prNb ... prNz, prNaa, prNab ... prNaz

Note the prefix ‘pr’ in all keys.

Package Options

```
\usepackage[keys={R,L,C},idx={{}},a,keys prefix = {pr},local defs]{tikzquests}
```

In this case, one will only get the following set of keys

- prRa, prRb ... prRz, prRaa, prRab ... prRaz
- prLa, prLb ... prLz, prLaa, prLab ... prLaz
- prCa, prCb ... prCz, prCaa, prCab ... prCaz

Note that, these keys will be defined only locally, when using `\Question`, `\tikzQuestion` or `\ftikzQuestion`.

Package Options

```
\usepackage[no alias, in review]{tikzquests}
```

In this case, no alias will be defined (the command `\defQuestionAlias` will be ignored), and when using `\tikzQuestion` (and similar) the question’s remarks (defined by `\defQuestion`) and annotations (from `\tikzQuestion`) will be printed. The `no alias` is specially useful when using the commands `\QuestionsFmtList` or `\QuestionsList`.

6.2 A More Complete Example

In the following code, an extra repository will be set (besides the default one) and two questions (a starred, text, and non starred, graphics) will be defined for each repository.

Defining Questions

```
% A repository name can be just about anything.
% the star makes sure 'Repo 2' is now the active/default one.
\defRepository*{Repo 2}

% quest names are even more flexible than a repository one
% the star implies this is a text one.
\defQuestion*{Quest A:1}{
  In the following circuit, assuming  $\beta \approx \text{\QuestVal{Beta}}$  and that  $V_{be} \approx 0.65V$ ,
    find the value of  $R_c$  such that the small signal gain is  $\text{\QuestVal{Gain}}$ .
}[That would be a question enunciate.]

%% Note the use of the macros \Ra, \Rb, \Rc, \Rd, \Vi, \Vbc and \Vo
\defQuestion[Repo 2]{Elect. 1a}{
  \draw
    (0,0) coordinate(A) to[V,invert,l=\Vi] ++(0,3) coordinate(V)
      to[R=\Ra] ++(2,0)
      to[C] ++(2,0) coordinate(B)
      -- ++(1,0) node[npn,anchor=B] (T1){}
    (A) -- (A -| B) coordinate(Ba) to[R=\Rb] (B) to[R=\Rg] ++(0,3) coordinate(C)
    (B) node[circ]{}
    (T1.E) to[R,l=\Rc] (T1.E |- A) -- (A)
    (T1.C) to[R,l=\Rd] (T1.C |- C) -- (C -| A) -- ++(-2,0) coordinate(X) to[V,l=\Vbc] (X |- A) -- (
      A)
    (T1.C) -- ++(1.5,0) node[ocirc]{} coordinate(k) to[open,v=\Vo] (k |- A) node[ocirc]{} -- (A)
    ;
}[this is a CircuiTikZ example]

%switching repositories
\SelectRepository{default}

% Note that, since this is a different repository, there is no name crashing.
\defQuestion*{Quest A:1}{
  In the following circuit, assuming  $\beta \approx \text{\QuestVal{Beta}}$  and that  $V_{be} \approx 0.65V$ ,
    find the value of  $R_g$  such that the DC level of  $V_o$  is equal to  $\text{\QuestVal{DC level}}$ .
}[Just for the sake of it.]

%% Note the use of the macros \Ra, \Rb, \Rc, \Rd, \Vi, \Vbc and \Vo
\defQuestion{Elect. 1b}{
  \draw
    (0,0) coordinate(A) to[V,invert,l=\Vi] ++(0,3) coordinate(V)
      to[R=\Ra] ++(2,0)
      to[C] ++(2,0) coordinate(B)
      -- ++(1,0) node[pnp,anchor=B] (T1){}
    (A) -- (A -| B) coordinate(Ba) to[R=\Rb] (B) to[R=\Rg] ++(0,3) coordinate(C)
    (B) node[circ]{}
    (T1.C) to[R,l=\Rc] (T1.C |- A) -- (A)
    (T1.E) to[R,l=\Rd] (T1.E |- C) -- (C -| A) -- ++(-2,0) coordinate(X) to[V,l=\Vbc] (X |- A) -- (
      A)
    (T1.C) -- ++(1.5,0) node[ocirc]{} coordinate(k) to[open,v=\Vo] (k |- A) node[ocirc]{} -- (A)
    ;
}[this is a CircuiTikZ example]
```

Once Questions are defined one can use them, for instance, using just the default parameter's values.

Questions Defaults

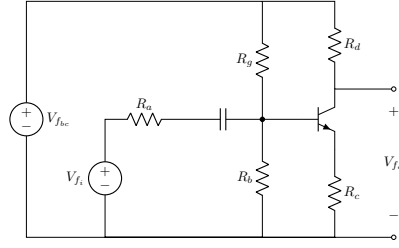
Choose one of the following two questions:

```
\begin{enumerate}
  \item \Question*{Repo 2}{Quest A:1}<just a last minute note about this>\par
    \ftikzQuestion(0.35)[Repo 2]{Elect. 1a}

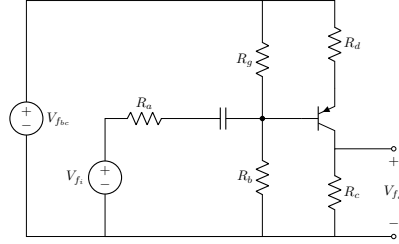
  \item \Question*[default]{Quest A:1}<just for the sake of it...>\par
    \ftikzQuestion(0.35)[default]{Elect. 1b}
\end{enumerate}
```

Choose one of the following two questions:

1. In the following circuit, assuming $\beta \approx \textcolor{red}{Beta}$ and that $V_{be} \approx 0.65V$, find the value of R_c such that the small signal gain is $\textcolor{red}{Gain}$.



2. In the following circuit, assuming $\beta \approx \text{Beta}$ and that $V_{be} \approx 0.65V$, find the value of R_g such that the DC level of V_{f_o} is equal to **DC level**.



Finally, one can use these same questions, setting it's parameters:

Questions Using Parameters

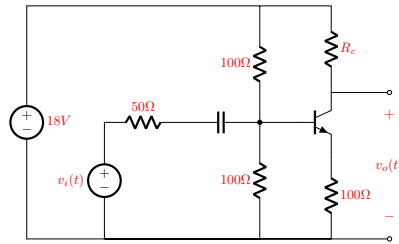
Choose one of the following two questions:

```
\begin{enumerate}
\item \Question*[Repo 2]{Quest A:1}[Beta=200,Gain=4,Ra=50\Omega]<just a last minute note about this>\par
\ftikzQuestion(0.35)[Repo 2]{Elect. 1a}[Ra=50\Omega,Rb*=100\Omega,Rg raw=\$100\Omega$,Rc=100\Omega,Rd=R_c,Vbc=18V,Vi=v_i(t),Vo=v_o(t),tikz keys={thick,red}] %%Note the use of the 'tikz keys' key...

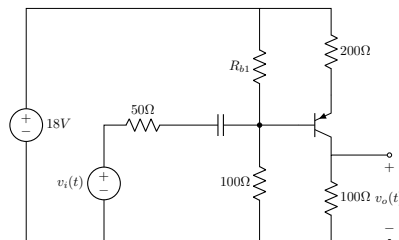
\item \Question*[default]{Quest A:1}[Beta=200,DC level=8V,Rg=R_{b1},Vo=v_o(t)]<just for the sake of it...>\par
\ftikzQuestion(0.35)[default]{Elect. 1b}[Ra=50\Omega,Rb*=100\Omega,Rg=R_{b1},Rc=100\Omega,Rd=200\Omega,Vbc=18V,Vi=v_i(t),Vo=v_o(t)]
\end{enumerate}
```

Choose one of the following two questions:

1. In the following circuit, assuming $\beta \approx 200$ and that $V_{be} \approx 0.65V$, find the value of R_c such that the small signal gain is 4.



2. In the following circuit, assuming $\beta \approx 200$ and that $V_{be} \approx 0.65V$, find the value of R_{b1} such that the DC level of $v_o(t)$ is equal to 8V.



6.3 Listing all Questions

To round it up, using the `\QuestionsList` (see 4.2):

L^AT_EX Code:

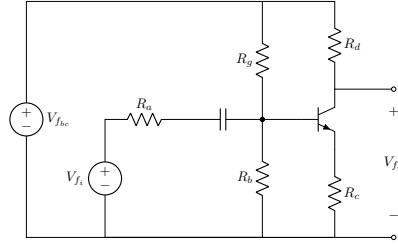
`\QuestionsList`

One gets:

Repository: **Repo 2**

non starred ones - TikZ graphics

Elect. 1a



Remarks: this is a CircuiTikZ example

starred ones - text/T_EX

Quest A:1

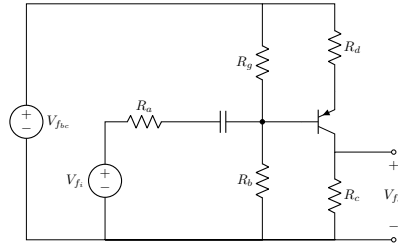
In the following circuit, assuming $\beta \approx \text{Beta}$ and that $V_{be} \approx 0.65V$, find the value of R_c such that the small signal gain is **Gain**.

Remarks: That would be a question enunciate.

Repository: **default**

non starred ones - TikZ graphics

Elect. 1b



Remarks: this is a CircuiTikZ example

starred ones - text/T_EX

Quest A:1

In the following circuit, assuming $\beta \approx \text{Beta}$ and that $V_{be} \approx 0.65V$, find the value of R_g such that the DC level of V_{fo} is equal to **DC level**.

Remarks: Just for the sake of it.