Aplikacja Rekrutacyjna na Wakacyjne Praktyki w Future Processing 2016.

Contents

1. High level explanation

1.1. The purpose of recruitment

The purpose of every recruitment is not to select those who are the “best” but for both sides - the company and the candidates - to select the counterpart who would match them the best.

In the course of creating the recruitment for internships in Future Processing we need to look at the company’s needs and calibrate the recruitment in such a way that we get the people most aligned with what the company currently needs from the potential hires.

This also means that every year we are recruiting slightly different people for internships, because the companies’ needs change and thus whom we are searching for also changes.

1.2. Whom are we searching for?

We are searching for software engineers. Or rather, potential software engineers. A software engineer in our understanding is a person who is able to perform the following roles:

- programmer / software developer (including working with the code they did not write themselves, transform specifications into solutions in code)
- quality assurance engineer (analyzing the domain, searching for errors, explaining the problem)
- a person able to work in team, able to balance priorities, doing things necessary to succeed and neither providing inadequate solutions nor over engineering and obsessing over irrelevant things
- a person being able to read English on communicative level, with less requirements on writing and even less on speaking.

1.3. Your potential benefits

By trying to solve the problem presented in this application you will be able to test yourself against areas of problems which would be a part of software engineer’s daily routine.

If you become one of the interns in Future Processing you will be a part of a group made of 15 people who, like you, have passed the application process. We will try to give you skills required to become a software engineer in the future.

This internship will not make you a perfect coder, it is not really 100% connected with writing code nor with particular intricacies of C# and .NET framework. This internship aims at helping you become a pragmatic problem solver who – when confronted with a difficult problem – will be able to solve it using code or other means (depends on what is the best for the team and the project). In short, what you are supposed to do at work.

1.4. The recruitment process

- stage I: you write the program corresponding to this application and send it with a document
- stage II: 40 people go to the workshop in Future Processing and teamwork with some additional aspects are being tested (you cannot really prepare for it; no technical skills are being tested here)
- stage III: 15 people are invited for internships.

1.5. Why this application?

Well, this particular application has several qualities:

- no solution exists in the Internet
  - so people knowing how to use Google do not have too much advantages
  - every year this application is different, so you can't even use past year solution
- the best possible solution is different for everyone; you can simply copy someone's solution
  - so people having friends do not have too much advantages
- this application tests what we are interested in and what we're searching for
  - so the final 15 should be on more or less close level
  - this document being informal is also an information for you; if you want very serious, formal stuff, probably those internships might be a slight nightmare for you
- you are able to decide if you want to even apply seeing how it looks like.
  - Recruitment is a two-way street: you need to fit us, but also, we need to fit your vision
  - so you do not waste your time and we get people we are interested in
- this type of recruitment and the application itself may be at least somewhat fun for you
  - and you won't have a feeling that you have wasted several evenings
  - so you have the feeling you have learned something even if you are not selected
  - so you can decide 'lol nope' seeing ~40 pages of document in English from the start.
- this looks much more like work you would have every day then writing a program from scratch (which very rarely happens in a standard work situation)
  - so you can take a stab on how would you fare in this particular kind of a problem

1.6.Why English?

As a wise man once said, “the most important language a programmer can learn is neither C# nor Java, but English”. We often work with the customers who communicate with us in English. A lot of documents and specifications are in English. Most tutorials and solutions to obscure problems we are meeting every day are in English.

Therefore, English language is required.

Therefore, recruitment application is in English.

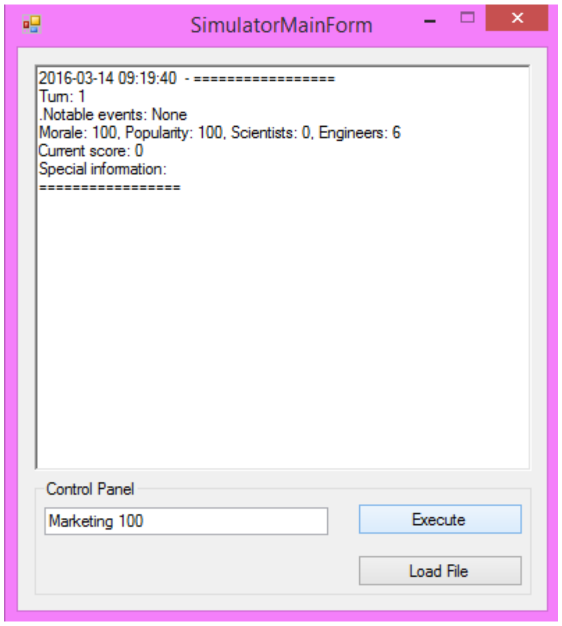
Our English is not perfect. Your English doesn’t have to be perfect either; it is enough for it to be communicative and understandable.

2. What are you given?

You are given the following:

- 1 (one) application document you are reading right now
  - which describes all the parameters and actions of the program, though it does not describe how exactly does the system work
- 1 (one) instance of a program which works in Visual Studio Community Edition

When you start an application you will probably see something like this:



```
2016-03-14 09:19:40 - =================
Turn: 1
.Notable events: None
Morale: 100, Popularity: 100, Scientists: 0, Engineers: 6
Current score: 0
Special information:
=================
```

Marketing 100

I have pressed the "execute" button and the above happened. I have used the 'marketing' action with intensity 100. Exact actions and their parameters shall be given to you in the later part of the document; what is important here is that you write the commands in the text box and fire them using the "execute" button.

The "load file" doesn't work. If you want to use it to create your own macros you will have to implement it on your own.

The story is, more or less this (at this stage you don't need more): you want to send a rocket to Mars. In order to do so, you need to research particular components first and then build it. And then launch it. By pressing "execute" you are ordering your people to do things. Because you want to launch the rocket to Mars.

3. What are you supposed to do?

Must fulfill all of the below:

- You want to send a rocket to Mars (system analysis, application analysis).
    - If you manage to send the rocket to Mars, your solution is good enough to get to the next stage of recruitment.
    - if you do not manage to send a rocket to Mars, your solution will not be considered.
    - In order to send the rocket to Mars you need to find a particular set of commands which will lead to researching the rocket components, building the rocket and in the end sending it to space.
- You want to write a report (clarity of thoughts, perseverance, ways to solve the problem)
    - if the document contains all the necessary information, your solution is good enough to get to the next stage of recruitment
    - if the document is written in Polish, your solution will be considered (because reading English is more important than writing English)
    - if the document is written in English, you will get extra points. It doesn't have to be perfect English; rule of thumb is "shall the person reading it understand what the writer meant".
    - If you do not write a document or one of key information is missing, your solution will not be considered.
- You must write logging of actions and parameters to a file
    - If the logging is present, your solution is good enough to get to the next stage of recruitment
    - If the logging is not present, your solution will not be considered.
    - If the logging is present but will not work as expected, your solution will be considered but you will lose quite a lot of points.

Should fulfill all of the below (not absolutely critical, but lots of points):

- You want to write an After-Launch Event (working with legacy code, software development)
    - if you manage to write an After-Launch Event, your solution is good enough to get to the next stage of recruitment
    - if you do not manage to write an After-Launch Event, your solution will be considered. However, After-Launch Event is a lot of points and unless the remainder of the report and the solution is very interesting you are unlikely to score higher than those who have implemented an After-Launch Event.

Would be nice if you fulfilled the below:

- try to maximize the score
    - the score itself is taken into account, but between different people maximum and minimum values of score will differ because of seed which is generated from your email. So if you have “500 points” and your friend has “550 points”, it does not mean that your solution is worse. It would be too easy to compare your solutions without the seed.
    - Do try to balance your real life priorities with recruitment priorities. If you feel you have “enough” score, if you feel you have done enough and if all the “Must” criteria are met (and you’re not having fun anymore), stop. It would be a shame to waste too much time for an application to internships where not everything depends on you. Stop especially if this is not fun for you.

So, the above is quite a lot to do. Consider this, though: our goal is to make as many people as possible quit at the first stage of recruitment. In an ideal world in the first stage of recruitment we would have 100-100000 people who wanted to write to the program, but in the end only 41 sent us the solutions. From those 41 we select 40 manually (and then those 40 go to second stage).

This application may be a lot of work for you, but this also means that we are minimizing human errors (the process of candidate selection in short time (less than a week between first and second stage selection) always has human errors) by reducing the amount of initial applications to consider manually. This will promote people who are persistent and who are willing to take a risk. And it promotes those who have the time in this particular time window, of course, but this is something we cannot really solve.

4. The context

So this is the year 2116.

Somehow we have managed to not end the world in the nuclear explosion, even if we tried. Somehow the Earth still exists and is alive. There have been many advances in terms of society, civilization and technology. We are working on the first space habitat on the Moon. And we are building a space station on Mars.

And Poland can into space.

You are a chief (CEO) of a company which is competing against some other companies for a contract to send a rocket on Mars. You are one of the modern companies, focusing not only on the pure profits but also on making human lives better. From the point of view of 2010, you are a CEO of a hopelessly idealistic company. From the point of view of 2116, you are the future.

Sending a rocket to Mars is just one of the difficult things to. There are also many things which need to be solved in the medical area, infrastructure, microcontrollers and intelligent human augmenting systems… And sending a rocket to Mars is a good way to test some of those things and to see whether some of those ideas your company has are possible to be materialized.

In short, it would be enough to send a rocket to Mars. It would not be enough for you. Being able to win the contract and send the rocket could also lead to advances in other technologies and sciences which would give you fame, recognition, money. And making the world better.

And you have the vision to make it happen.

You have won a preliminary contract and got a lot of budget money (most of it from European Union, space program). You have a core, strong team. You have some facilities and some ideas.

You can make it happen.

Your competitors are squabbling against each other, while you look at what is to be done and aim for the stars. Well, Mars, actually. Ignore the competitors. Go out there and win.

As usually, your only opponent is yourself.

5. The domain

5.1. The overarching view

In the bullet points, because bullet points are awesome.

Overall:

The whole program can be summarized in the following sentence:

"You influence the system using actions in such a way that you maximize the parameters to increase the progress to generate proper events to send a rocket to Mars. Then you aim at maximizing the score and expand your solution so you do not only send a rocket to Mars but also make other good things happen."

And that's basically what you are to do.

Actions:

- There are the following actions you can take:
  - `Focus Science`
  - `Focus Production`
  - `Perform Work`
  - `Hire People`
  - `Fire People`
  - `Marketing`
  - `Slower Time`
  - `Launch Rocket To Mars`
- Every action can have zero or more parameters. Look at particular action's description to see.

Parameters:

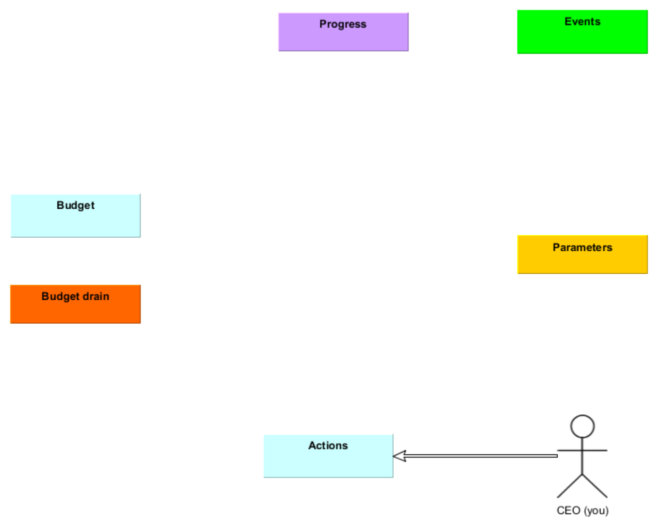
- Parameters have an informative role. Parameters in the system are:
  - Score
  - Budget
  - Budget drain
  - Morale
  - Popularity
  - Scientists
  - Engineers
- Feel free to look at particular parameters to understand their meaning and their purpose

Progress:

- Progress has a main role in this application. It represents the accumulated sum of resources and potential to either construct something or make a scientific breakthrough (discovery). In general, you want those numbers to go up.
  - Primary research progress
  - Primary production progress
  - Secondary research progress
  - Secondary production progress
  - Cargo production progress
- It will be explained properly in an appropriate part of the document.

5.2. Description of subsystems you can use

Being a CEO of a company trying to send a rocket to Mars means you need to understand the interrelations between particular parameters and things you can do. A very short explanation in diagram form:

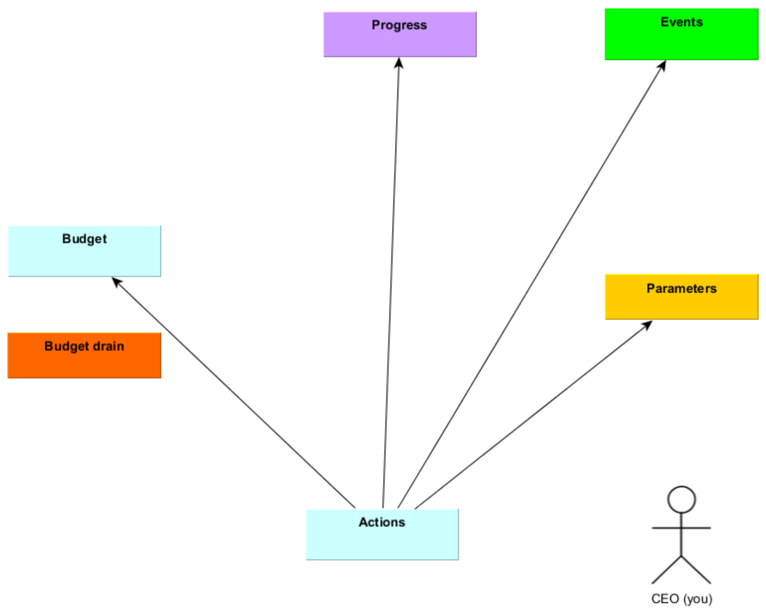


I am perfectly aware that this is not what CEO is supposed to do in real life, but this is not reality. So what you actually do, dear reader, is order some actions to happen and they will influence the system in some way.

The only thing you are able to actually influence from your control panel is ordering actions by pressing buttons. There exists an engine in the code to which you have no access (no normal access) and it transforms the strings you sent as actions into particular things which happen. So, first word to remember is “action”. What you order to make things happen.

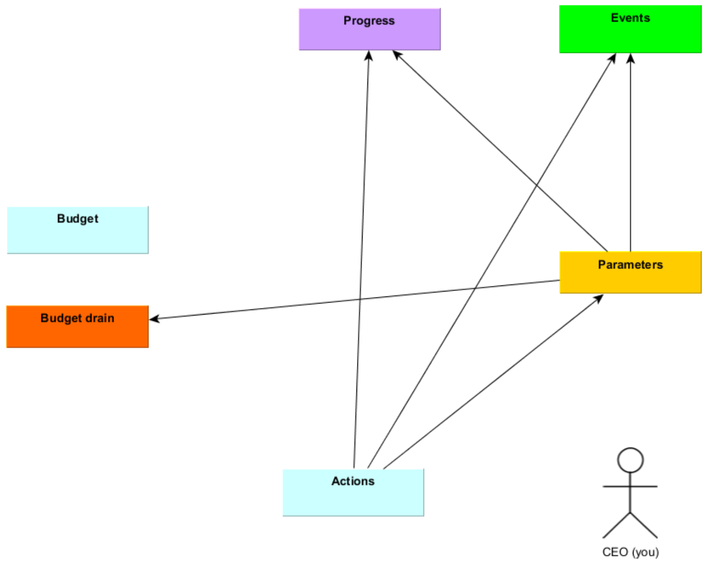
Every action takes one month out of 24 months you have.

And what happens when the action is fired?



In short, actions are your main way to manipulate the system. I would say it is your only way to manipulate the system (aside writing the code for an event). An action can influence parameters, it will cost some budget, it may fire some events and some actions make the progress happen. You will have actions explained in the following part of the document, but not every interaction and not every implication of those actions - that's your work.

Let’s look at the relation between an action and a parameter:

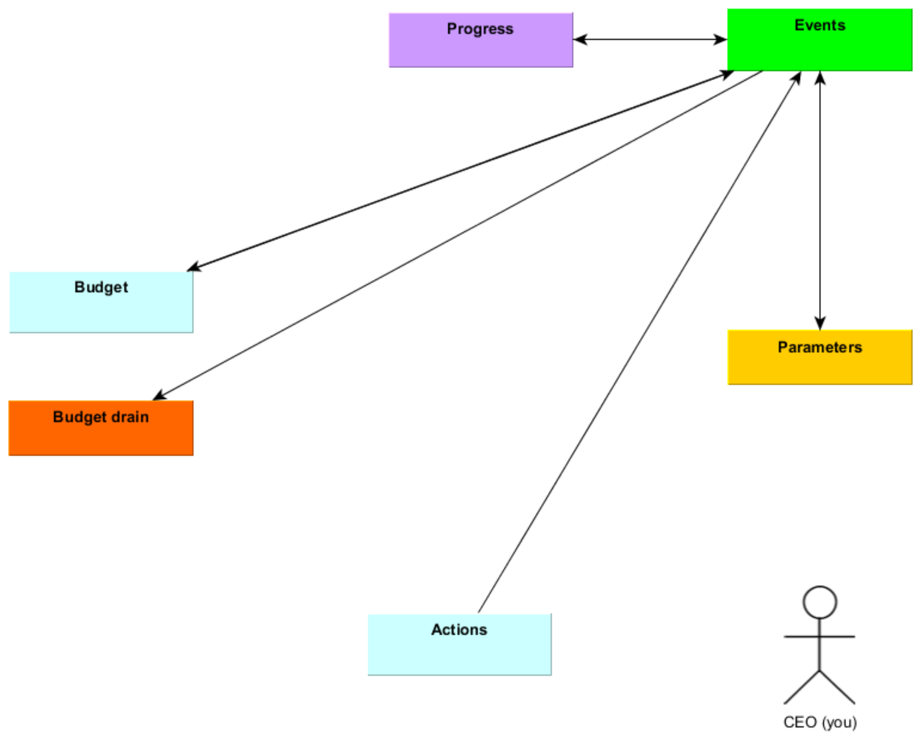


An action does influence the progress, but also do some parameters. So to make the progress happen, you need to take into account not only the action itself; you need to see how the parameters influence the power of action.

Also, it isn’t that simple that an action in itself fires events. What really happens is that the combination of parameters having particular values with an action will cause an event to happen. What and how does it happen is up to you to discover.

Parameters, especially “engineers” and “scientists” may lead to budget drain. Budget drain is a value which is subtracted from the budget every month (every action). So it is a tax, in a way. If you run out of budget, something terribly bad should happen.

Events and how they are generated:



As you can see from the picture above, events can be generated by absolutely everything (well, mostly everything). And also they can influence almost everything. Notably, they cannot fire actions even if they can do things as if actions were fired.

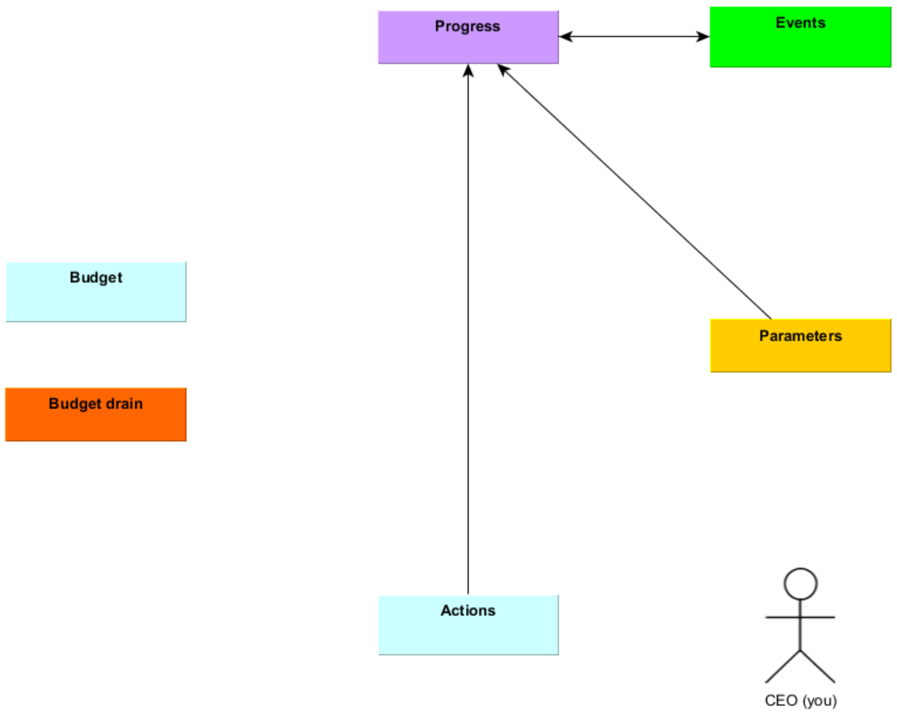
Events always influence the score.

I will give you a hint: one of the events is called "engine built" and it represents the act of building a particular engine for the rocket to Mars. Events are what is going to enable you sending the rocket to Mars.

You will have to implement one event, so-called "After-Launch Event" (in short, ALE). This will greatly improve your score and ALE will be specified later.

Discovering events and basing your strategy around them may be beneficial in terms of your score. It may also be a waste of time; it depends how much time you will invest in them.

The progress and why it is not one of parameters:



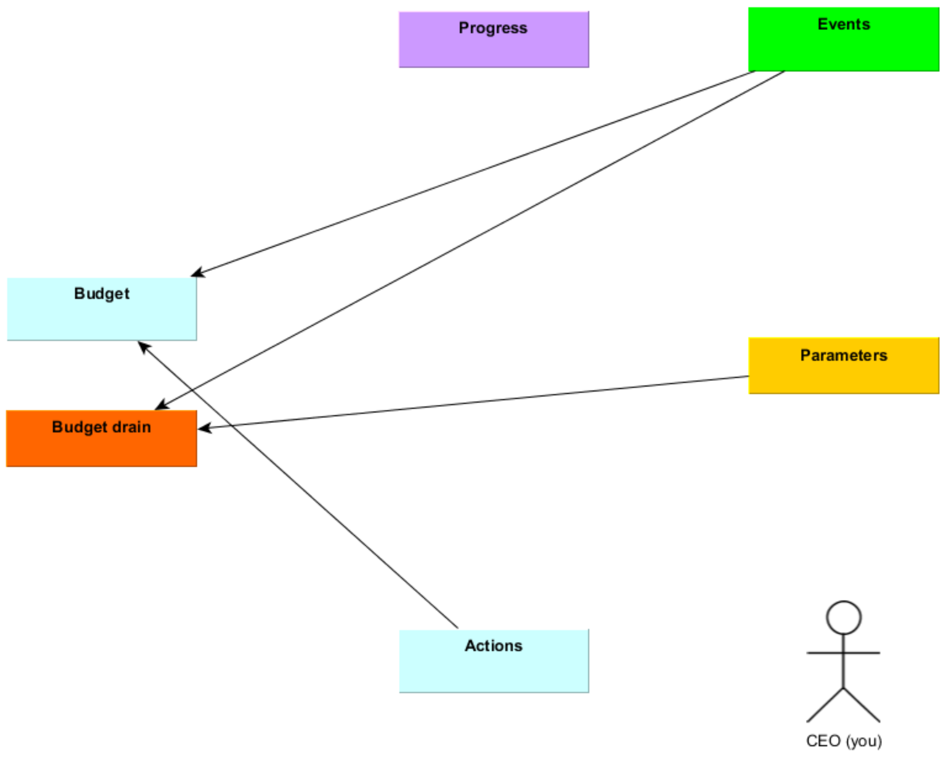
You have five progress pools. The progress is the only way you can manage to send a rocket to Mars, make some scientific breakthrough or construct something. In short, all the events required for sending the rocket to Mars will happen when progress goes up.

As you can see, the only way to get progress higher is to fire appropriate actions having properly fine-tuned parameters and sometimes have some events.

You could say that "parameters" are what support "progress". Getting progress higher leads to getting rocket to Mars faster and getting some progress events. But without particular parameters some events will not happen even if progress is met.

Something for you to find.

The budget, or “money, why are you so important”.



In general, budget goes down; not up.

Every action will make one month pass; every month the value of budget drain is subtracted by the value of budget.

Every action may have some fixed cost subtracted from the budget when an action is fired.

Depending on the value of some parameters budget drain can be increased or decreased.

Some events may increase the budget some may further decrease it same with budget drain.

Like in life, don’t run out of money.

5.3. Description of components

5.3.1. Actions

- Focus Science

Syntax	focus_science type intensity
Example	focus_science primary 1
Method Parameters	Type: **primary** or **secondary**. Primary is connected with researching things towards sending a rocket to Mars. Secondary is connected with researching things towards bettering the general human state. Intensity: number. Intensity means basically “how much energy do we put towards it”. Higher means more.
What it does	This action represents focusing on researching things. It influences science progress (research progress), be it primary or secondary, as selected.

- Focus Production

Syntax	focus_production type intensity
Example	focus_ production primary 1
Method Parameters	Type: **primary**, **secondary** or **cargo**. Primary is connected with researching things towards sending a rocket to Mars. Secondary is connected with researching things towards bettering human state. Cargo is required for actually sending secondary production to Mars. Intensity: number. Intensity means basically “how much energy do we put towards it”. Higher means more.
What it does	This action represents building things which were researched before. It influences production progress, be it primary, secondary or cargo.

- Perform Work

Syntax	perform_work intensity
Example	perform_work 1
Method Parameters	Intensity: number. Intensity means basically “how much energy do we put towards it”. Higher means more.
What it does	This action represents a normal work month, without focusing on anything in particular. Should influence all progresses.

- Hire People

Syntax	hire_people type intensity
Example	hire_people engineer 1
Method Parameters	Type: **engineer** or **scientist**. Engineer influences production. Scientist influences research (science). Intensity: number. Intensity means basically “how much energy do we put towards it”. Higher means more.
What it does	This action represents hiring people of particular type in order for them to do stuff for the company.

- Fire People

Syntax	fire_people type intensity
Example	fire _people engineer 1
Method Parameters	Type: **engineer** or **scientist**. Engineer influences production. Scientist influences research (science). Intensity: number. Intensity means basically “how much energy do we put towards it”. Higher means more.
What it does	This action represents firing people of particular type in order to get some savings. Simplified, I know.

- Marketing

Syntax	marketing intensity
Example	marketing 1
Method Parameters	Intensity: number. Intensity means basically “how much energy do we put towards it”. Higher means more.
What it does	This action represents engaging with the media, showing your product and basically showing the public your greatness.

- Slower Time

Syntax	slower_time intensity
Example	slower_time 1
Method Parameters	Intensity: number. Intensity means basically “how much energy do we put towards it”. Higher means more.
What it does	This action represents giving people some time off, giving them some opportunities to blow off steam and to relax.

- Launch Rocket To Mars

Syntax	launch_rocket_to_mars
Example	launch_rocket_to_mars
Method Parameters	None
What it does	This action finishes the simulation. It is an act of sending a rocket to Mars, the last thing you will do before you will reap the fruit of your soil.

5.3.2. Parameters

- Score

Something on meta level you want to have maximized. More generally means better. Irrelevant from the point of view of the simulation, having this high is the goal of yours.

- Budget

The amount of money and resources you can pour into sending a rocket to Mars and bettering the current human state of affairs.

- Budget drain

The amount of money subtracted from the budget every month (every action)

- Morale

How much your employees love you (or hate you).

- Popularity

How much the world outside your company loves you (or hates you).

- Scientists

How many scientists work for you. Scientists research stuff.

- Engineers

How many engineers work for you. Engineers build stuff.

5.3.3. Progress

- Primary research progress

Things we have researched which allow us to send a rocket to Mars.

- Primary production progress

Things we have built which lead towards assembling a rocket to be able to actually send it to Mars.

- Secondary research progress

Things we have researched which allow us to better the humanity; some loose ideas and things we could use the space program to test.

- Secondary production progress

Things we have imagined using secondary research are being built using this progress.

- Cargo production progress

Basically, cargo. A set of containers and special boxes which are used to transport the result of secondary production to Martian base.

6. How to start?

6.1. Overall

You may have no experience with working on this types of problems. Furthermore, you may have no experience working with Visual Studio. That is okay. This application is calibrated in such a way that you do not have to have all of that. They help, but are not necessary.

Read the instructions thoroughly. Follow those instructions. Read the documentation. Do what is expected of you and nothing more. Do not sacrifice the time you are not willing to lose. Follow the sequence denoted in this numbered list. Remember, that it is better to deliver a 1500-points document then to not deliver (because of perfectionism and lack of time) a 9999-points document.

Read the document you have written. Check if everything important is there. Double check it.

Run your version of the application. Make sure it builds. Make sure it works on a machine which is not your machine. If you need some special library and you do not include it properly, chances are we do not have that library. Reconsider if you want to use it.

When everything is correct, send us the solution.

Do not forget about the boring details – great people have failed because of a small, stupid detail in a wrong moment. Do not be one of those people.

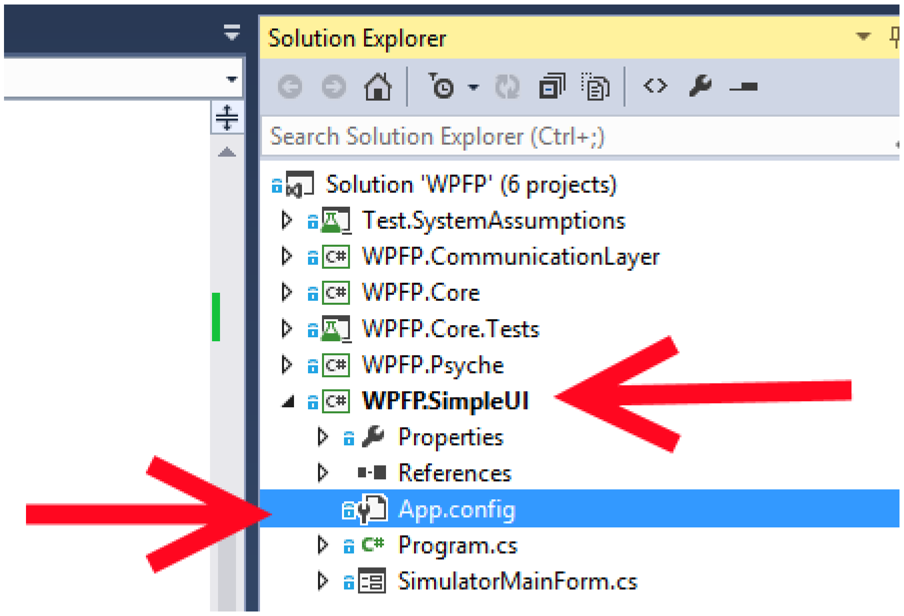
6.2. Configuration of an application

The absolutely first thing you need to do is to input your identifier into the program. Your identifier is your email; and this email will actually be used as a seed - according to your email your starting parameters will vary and costs of some actions (or points resulting from some events) will change.

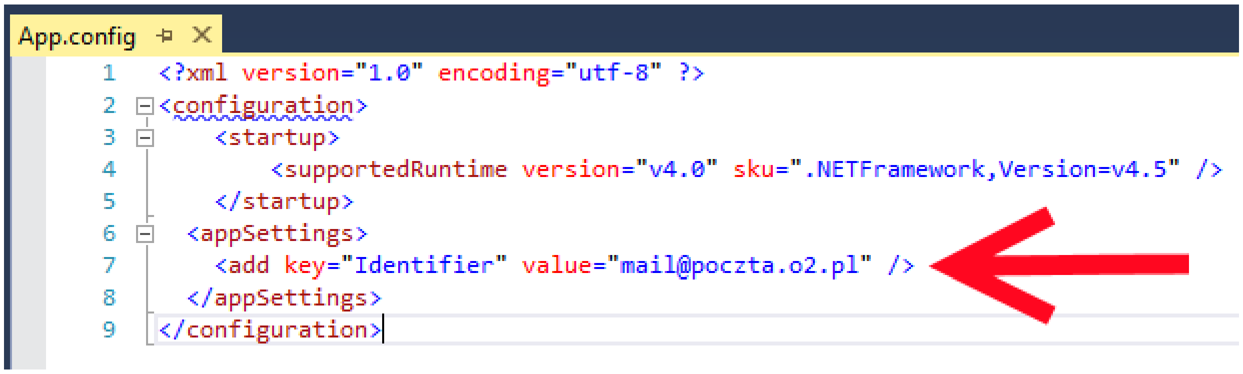
When we evaluate your program we will look at the identifier to make sure that the seed we are testing your application with is the same seed you had. That way we can normalize the score between people having different seeds.

You do not have to give your primary email or an email you use in recruitment process. The email you are going to use may be fictional - we won't ever contact you using that particular email. We do want that email to be unique, though (otherwise what's the point of seed).

As the best possible solution changes with seed, the absolutely first thing you want to do is to set a proper identifier. To do that, open Visual Studio solution and find your starting project:



Having opened the App.config file you need to change the record with identifier:



```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
    </startup>
  <appSettings>
    <add key="Identifier" value="mail@poczta.o2.pl" />
  </appSettings>
</configuration>
```

If you see something different, do not worry; I'm probably using an old version. What is important is the **key** "identifier" and the **value** "mail@poczta.o2.pl".

Change the **value** into an email you want to have C degenerated from.

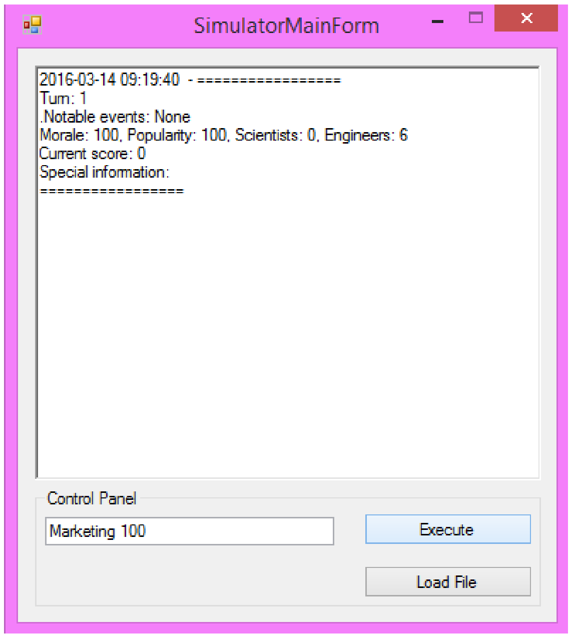
For example:

„mail@poczta.onet.pl" -> „krwawy.kurczaczek.grozy@buziaczek.pl"

And that way your application will be configured. Feel free to go to the next stage.

6.3. Experimenting phase

After configuring the application, run it. You’ll see something looking more or less like this:



```
2016-03-14 09:19:40  - =================
Turn: 1
.Notable events: None
Morale: 100, Popularity: 100, Scientists: 0, Engineers: 6
Current score: 0
Special information:
=================
```

Marketing 100

Look at the description of actions and try making some things happen. Basically, try understanding how the system works and try to get the rocket to Mars.

This stage ends when you have sent the rocket to Mars and you have any sequence of steps which allows you to do so.

6.4. Logging implementation

One thing we expect from the application and from the report is being able to see what you have actually done in terms of sequence of commands combined with their intensities, and the results of those. You need to write it in the final document, actually.

This should look similar to this:

```
===================
Month: 1
Notable Events: EngineResearched
Popularity = 30
Morale = -20
Budget = 2100
…
===================
```

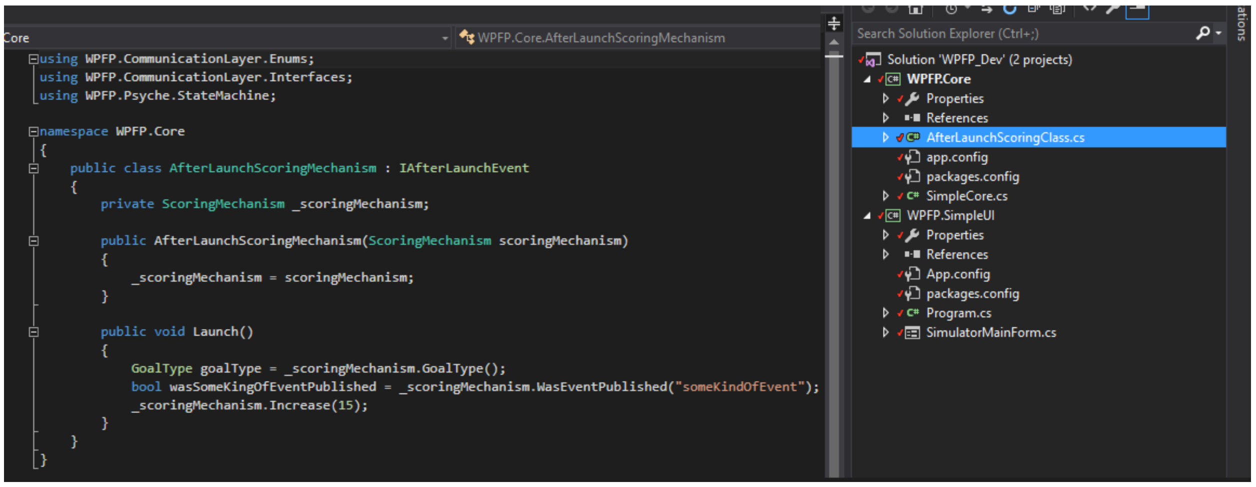
Of course what you see right now in the control panel (what is displayed to you in UI) does not have all required information. The information is present in the code, however. So your goal will be to take all interesting information and finally dump it into the file.

Note the hardcoded filepath "C:\pinkunicorns\earthworm.txt" is not really acceptable. The log should be generated in the project folder, somewhere. And it would be wise to write in the report WHERE it is created; so if I run your application in E:\program files\iwanttogotomars\yoursolution, the log should be generated somewhere in 'yoursolution' folder in that path.

How to do this filepath magic is googlable, in case you have never done it before.

6.5. After-Launch Event implementation

Now your rocket has been sent to Mars, time to reap what you have sown. In the project named WPFP.Core there exists a class “after launch scoring effect”:



Core
WPFP.Core.AfterLaunchScoringMechanism
Search Solution Explorer (Ctrl+;)
ations

```
using WPFP.CommunicationLayer.Enums;
using WPFP.CommunicationLayer.Interfaces;
using WPFP.Psyche.StateMachine;

namespace WPFP.Core
{
    public class AfterLaunchScoringMechanism : IAfterLaunchEvent
    {
        private ScoringMechanism _scoringMechanism;

        public AfterLaunchScoringMechanism(ScoringMechanism scoringMechanism)
        {
            _scoringMechanism = scoringMechanism;
        }

        public void Launch()
        {
            GoalType goalType = _scoringMechanism.GoalType();
            bool wasSomeKingOfEventPublished = _scoringMechanism.WasEventPublished("someKindOfEvent");
            _scoringMechanism.Increase(15);
        }
    }
}
```

```
Solution 'WPFP_Dev' (2 projects)
  WPFP.Core
    Properties
    References
    AfterLaunchScoringClass.cs
    app.config
    packages.config
    SimpleCore.cs
  WPFP.SimpleUI
    Properties
    References
    App.config
    packages.config
    Program.cs
    SimulatorMainForm.cs
```

You can see some simple implementation of the launch method. Your goal will be to implement the proper scoring method according to specification given.

The ALE needs to follow the algorithm of:

Dictionary: “event” == “event has been present at least once in the history of the run”

if CockpitResearched event: score = score + 1000
if RocketAssembled event: score = score + 1000

If GoalType is FutureCity then for events: UniversalBattery, MoreEfficientTransportationTakesWorldByStorm : score +1000 each

If GoalType is HealthCare then for events: CleanerFuelResearched, SafeStimulantsSold: score +1000 each

If budget < 1000 then score = score + 1000
If popularity > 50 then score = score + 500
If budget < 500 and engineers + scientists < 40 then score = score + 500
If budget < 250 then score = score + 250
If budget < 50 then score = score + 250
If engineers + scientists < 20 then score = score + 500

This stage ends when you write code which meets the specification above. To make sure, look at “what do we want from you”.

6.6. Even more experimenting

If you have finished an “After-Launch Event implementation” or if you decided you don’t want to do it, it is time to start experimenting with maximizing the score.

Now you can focus on research secondary and production secondary and cargo secondary. Try to generate as many events as possible. Try to understand when the events are generated and what do they rely on. Even those events which do not lead towards the best possible solution our worth it; the more events you identify in the report, the better.

I will give you a hint:

- the most influential event in terms of points is sending a rocket to Mars
- the second most influential event is a proper After-Launch Event implementation

Everything else is a nice bonus.

This stage ends either when you are slowly running out of time or when you stop having fun.

6.7. Creation of the document

Having some observations, some thoughts and ideas you are able to start writing the document. Note that the document has several questions posed for you to answer. If you do not know, write “I do not know”. It’s okay.

A well-formed document is not too long (you do not want to waste time writing something which doesn’t benefit you). It should contain your thoughts and observations. There are no “right answers”; only different approaches. There is no checklist we are checking ‘he used a buzzword and she used a framework – checked’. We are trying to understand how you think and how you solve the problems. Any solution which works is good enough from our point of view.

Yes, you may have no idea how often a technical person has to write a document. An example being this document you are reading right now…

7. What do we want to get from you?

7.1. Overall

We expect one application and one document from you. The application and the document are both important.

The way we give you final points goes as follows:

```
totalPoints = scoreFromApplication * applicationModifier *
documentModifier
```

where:

scoreFromApplication = whatever score you got in your final solution, after being normalized by seed. In short, whatever your program displays as score but after normalization.

applicationModifier = how much we like your code. Between [0-100%] range.

documentModifier = how much we like your document. Between [0-100%] range.

As the numbers and application are set in such a way that the things we want to see the most are scored higher than the seed-dependent things, do not obsess over every single point of score.

The above gives us a list. An ordered list of all people who answered the application. As in reality we expect most applicationModifiers and documentModifiers to be quite similar, we will be able to easily divide all answers into the groups of “they did everything in satisfactory way”, “they did most things in satisfactory way or they did everything in less satisfactory way”, “they did not do enough”. We will look at the first two groups’ solutions and from those we will manually pick 40 people.

Note that being positioned higher on the list does not mean being picked. But being positioned higher on the list greatly increases the probability of being picked.

7.2.An application

7.2.1. What do you have to deliver in application

What we expect is, basically, an application which compiles and runs and which does things it is supposed to do.

Namely:

- we expect you to write the logging of the text to a file
    - every simulation starts at month number one and ends at some point. We expect to see the log between the first month and whatever finished the simulation
    - by "the log" I mean the text representing all the information about the state of the system, notable events, whatever happened and can be useful in terms of understanding what actually happened

```
==================
Month: 1
Notable Events: EngineResearched
Popularity = 30
Morale = -20
Budget = 2100
…
==================
```

-
    - the format of this text is up to you. It is to be readable, no matter how you understand 'readable'. It should have all information you deem useful.
    - The log should be created at the end of the simulation; no matter if the simulation finished because of time or because of some events.
    - The log should be generated in the project folder
    - This log should be included in the document in an appropriate section

- We expect you to write an After-Launch Event
    - it needs to match the specification
    - it needs to add the score properly

7.2.2. How is the application scored

The application is written in such a way, that the following is true:

- Most points - no matter the seed - will come from the launching a rocket to Mars.
- Second most points - no matter the seed - will come from implementing the After-Launch Event.
- Only then the events and the seed kick in.

This shows you the priorities and what to focus on particular moments.

If the application does not compile or is does not run, we will not look at anything. We are very unlikely to attempt to make it work. It is your responsibility to deliver us the application which runs.

After we receive your solutions we will score your code and things you have done. We will look at the following:

- The presence of a unique identifier (20% points).
- The completeness of the solution (35% points)
- The readability of the code. (up to 10% points)
    - Code is in English
    - Code is consistent with other code already written in the application
    - Is the naming proper? Are the names of the functions / variables / classes self-commenting?
    - How easy would it be for a third party to work with this code?
- The structure of the code. (up to 10% points)
    - If applicable, would this code be testable?
    - If applicable, DRY & SOLID – are they applied here?
    - Proper code in a proper place.
- A good solution. (up to 10% points)
    - Lack of bugs in the code you have generated.
    - Understanding the domain / the situation.
    - Doing what is to be done in instructions and not doing things beyond that.
    - Use of a proper tool (code, algorithm…) for an appropriate problem.
- Extras. (up to 10% points)
    - Automated tests, if applicable.
    - Interesting solutions, but ones which are not overengineered.
- Something we deem sufficiently interesting (up to 5% points)

So if you deliver the code which does what it should and contains a unique identifier you get 55% points already. Readability, structure and properness of solution would yield you 85%. Cool ideas and solutions are worth 15%. Therefore it is better to focus on the basics; we expect most applications to score between 65 and 80% here.

7.3. A document

7.3.1. What do you have to deliver in document

Basically, we expect a readable document containing your thoughts and observations.

Truly, <u>it is enough to write 2 to 3 sentences per question</u>. You can write in Polish, but you will score higher if you write in English.

1. What you are your inclinations toward software engineering?
    a. What was the most fun for you while working with this application? Why?
    b. What was the least fun for you while working with the application? Why?
    c. Where did you spend the most time? Why?
    d. What are you the most proud of (while working with this)? What is your greatest achievement in terms of making this solution happen?
    e. What would you do differently if you started doing this again?
    f. What was the most difficult thing while working with this application? Why?
2. How did you arrive to the solution?
    a. How did you get to the solution, what steps did you take, where did you backtrack… In short, please write the “story” how did you get here (2-3 sentences are enough).
    b. How did you explore the application?
    c. How did you maximize the score?
    d. What type of hypotheses did you make and how did you verify them?
    e. Do you have an interesting observations you would like to share? Any thoughts about the process of solving this problem you would like to write?
3. What was the solution?
    a. What was your identifier (from which we can generate seed)?
    b. What was the solution you deemed to be the best (sequence of actions and intensities)? In short, give us a script (log; yes, log goes here).
    c. Have you found any bugs in the code? How severe were there if you have? Why?
    d. What types of events have you encountered and when are they fired?
    e. How do you understand the system to work (what influences what; write 3-5 things which are not in the specification and which you deem the most important/interesting)?
    f. How do you think the score is calculated? Can you give your approximation of a formula? (use “After-Launch Event score as ALE_score to not repeat stuff from documentation)
4. How would you improve the recruitment?
    a. What would you change in the application itself (in the set of tasks you are supposed to do)? Why?
5. Anything you want to write to add which is not in the questions (optional)?

7.3.2. How is the document scored

Basically, we score function over form and having our questions answered. What you write inside the answers cannot really be scored; it is your observations. So if you answer all the questions in readable English, you will probably get the maximum points.

Specifically, we're scoring the following:

- The presence of a unique identifier (25% points)
- Readability and the structure of the document itself (up to 10% points)
- Completeness of the document (answers to the questions above) (40% points)
- Your approach, the methods used, level of analysis (in short: things you write answering the questions)… (up to 10% points)
- Interesting things we did not think about (up to 5% points)
- The document is written in English (10% points)

In short:

- Answer the questions in satisfactory way (it's enough that we see that you try to answer those questions as well as you can, not dodge them) and do not forget about identifier - you get 65% of points.
- Make a document readable and properly structured - you have 75% of points.
- Write in English - you have 85% of points
- If you use an approach we deem very interesting or an approach we deem to be very appropriate for a solution, you can get 95% of points
- Astonish us with anything, do or write something we haven't thought about - 100% points.

We expect most applications to be in 75-85% range. Do not aim at more; it is important to deliver a working solution over not delivering a perfect solution and you don't really <u>know</u> what we deem to be "appropriate" or what will astonish us. Don't waste your time on things out of your control.

8. Practical hints

8.1. Congratulations

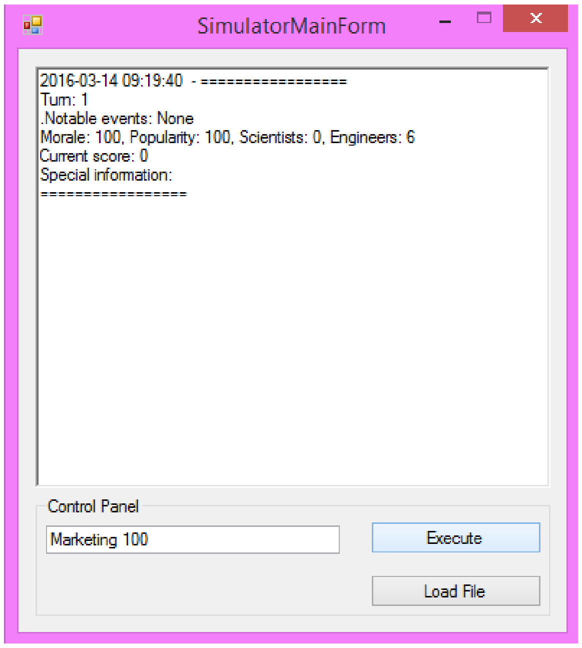
No matter if you have stumbled upon this section by accident or if you were searching for something like this (or you did not start working before reading everything), you are here.

This section will give you some practical hints and tips how to work with this application.

Also, note in the report that you have read the practical hints section. This will tell us you have read and remembered.

8.2. Modify the control panel display

In the very beginning, you are given something like this:



```
2016-03-14 09:19:40  - =================
Turn: 1
.Notable events: None
Morale: 100, Popularity: 100, Scientists: 0, Engineers: 6
Current score: 0
Special information:
=================
```

Yet as you can see if you have read this document not every parameter here is described. Notably, you do not see budget (nor budget drain). You do not see the progress, any of it. You lack most of the important information.

If I got something like this the first thing I would do is to go to the code and look at the moment the string is displayed on the text box. Most probably the class contains all important information but this information is not displayed.

So you want to modify the output string in such a way you will get all important information in the format you consider readable.

8.3. Implement the “Load File”

When you press the “Load File”, nothing really happens.

Well okay, depends on our mood: either nothing happens or you get an exception.

Imagine how much time it will be to write:

“focus_science primary 1”
“focus_science primary 1”
“focus_science primary 1”

every time we want something to happen. And every typo you make will not enter the engine (because it is brittle by the very design; we want it to hurt you so you will implement Load File).

Wouldn’t it be amazing to have a text file which you can read and in which you can simply write sequences and see things happen? To write down the scripts and juggle them to see the best script possible? Perhaps even make scripts fight each other for the highest score possible?

Okay, maybe “Script Gladiators” is not the game we’re working on. But if I were you I would definitely start from working with text files, not manually input text in the text boxes. Because it would be faster and more efficient.

8.4. Focus on sending rocket to Mars first

You have a lot of parameters, lots of actions and many things you can play with. Yet there needs to be purpose in all of that because you have limited time.

From my experience the most important thing is to first send a rocket to Mars and first do what there is to be done and then try to expand the solution in such a way that the score is maximized.

In student’s terms, first make sure you get a ‘3’. Then aim at ‘5’. If you run out of time, you will have at least ‘3’, while those who were aiming at ‘5’ from the very beginning may have better solutions than you – and so what if they did not deliver anything at all?

Note that this program is tricky. It does not have exact criteria when stuff is done. It requires discipline on your part, otherwise you will slave overnight and you will still not be satisfied with the result.

Trust me, I know.

8.5. There may be some bugs in the code

I know, inconceivable. A bug is any discrepancy between the documentation and how the program works. Also, a bug is any malfunction of the code. And this codebase is brittle.

When you encounter a bug, be happy. There is a special place in the report you can write the bugs you have encountered. Remember to explain why you think what you have found is a bug and how severe is this bug. It's extra points, after all.

8.6.Remember that there are no "wrong solutions"

You have solved the application. Perhaps you have dutifully clicked through every possible permutation. Maybe you have tried to brute-force it. Maybe you have decompiled it. Maybe you outsourced it to younger siblings…

Well, maybe the last one is a wrong solution. But everything you do on your own is fair game. If you think you have done something unethical, think again. As long as you are working on the solution alone, without the help of others, there is nothing you have to be ashamed of. In "real life" the result is important; no matter your approach, as long as it is ethical and legal it is perfectly okay.

So feel free to write your mental processes and your ideas in the document. There are many interesting ways how to solve this particular application and how to maximize the score. You are not using a "wrong solution", really. You are using a solution which fits you the most, one you deemed the most appropriate for the situation.

No need to hide what it was. Things you might consider "bad" might be considered "interesting" and "innovative" by people who will read your document.

Besides, would you really like to be an intern in the company where you would have to hide your approach to the problems?

8.7.Remember to have fun

However strange it sounds, having fun is important. At one point you will get bored with this application. Remember the 80-20 rule (80% of the result take 20% of time. Last 20% of result take 80% of time). This is not a real world problem; it is not something people are going to die if you do not put everything into.

I can’t guarantee that other people will not obsess over every detail. But because the score is dependent on seed and because the application is time-consuming and pretty difficult for an internship application, it is very easy to put too much time into this one and it does not give you 100% probability of being accepted. Investing 20 hours is one thing, investing 1000 hours without being able to guarantee the success is from my point of view unacceptable and is a wrong allocation of your time and energy.

So when you are getting bored with this application or when it is not fun anymore, stop.