CONTENT OF YOUR BASH .SH SCRIPT

3 parts, from the top of your .sh file:
1. '#!/bin/bash'
2. The SBATCH directives to Slurm
3. The job steps (example provided below: the job is to run an R script)

2. SBATCH directives to Slurm

```
#SBATCH --account=<myAccountName>      # Account to be charged for resources used
#SBATCH --job-name=<myJobName>         # Job name
#SBATCH --nodes=1                      # Nb of nodes to assign to this job (= on which to distribute the Tasks)
#SBATCH --mem=<MB>                     # Memory required per node [in MB]
#SBATCH --time=10:00:00                # The time the job will take to run (d-hh:mm:ss).
#SBATCH --mail-type=ALL                # Mail events (NONE, BEGIN, END, FAIL, ALL)
#SBATCH --mail-user=<…@…>              # Email address to send job-related status
#SBATCH --output=</path/filename>.out  # Set file in which to store output (default = creates `slurm-########.out` in working directory)
#SBATCH --error=<filename>             # Set file in which to store job error messages
```

Optional:

```
#SBATCH -n x    Number of CPU cores you are requesting x = 5,10,20,etc
#SBATCH --ntasks=1                     # max nb of tasks executed in // (across nodes)
#SBATCH --ntasks-per-node              # nb simultaneous processes to run on each node
#SBATCH --cpus-per-task=1              # nb of CPU cores required per task
#SBATCH --mem-per-cpu=<MB>             # Memory required per allocated CPU core
#SBATCH --output=<filename>            # File in which to store job output
#SBATCH --array=<indexes>              # array range (ex: '--array=21-40%10')
```

Either --ntasks, or --ntasks-per-node & --nodes

<u>Equivalent syntaxes:</u>

- The account name for the job
  - #SBATCH --account=<myAccountName>
  - #SBATCH -A <myAccountName>
- The job name

- #SBATCH --job-name=<myJobName>
- #SBATCH -J <myJobName>

• set output & error files (default: creates `slurm-###.out` in wd)

- #SBATCH --output=<filename>
- #SBATCH --error=<filename>
- #SBATCH -o array_%A.out
- #SBATCH -e array_%A.error

• Total run time limit (d-hh:mm:ss)

- #SBATCH --time=10:00:00
- #SBATCH -t 10:00:00

• total number of tasks

- #SBATCH --ntasks=1
- #SBATCH -n 1

• nb of nodes

- #SBATCH --nodes=1
- #SBATCH -N 1

3. The job steps. Ex: R CMD BATCH

In this example, the computing 'job' is to run an R script.
We will use the R tool `BATCH`, which is designed to run R in batch mode.

The general form will be:
module load R
R CMD BATCH <options> <infile> <outfile> <args>

<options>
default options are `--restore --save`
--save : saves all the objects in the workspace (i.e. all the objects created while the script was executed) into a hidden file `.RData`
--no-save : doesn't save data sets at the end of the R session
--restore : loads the content of the file `.RData` in the directory where R was started
--no-environ : doesn't read any user file to set environment variables
--vanilla : combines `--no-save`, `--no-environ`, `--no-init-file`, `--no-restore`
--silent : don't print out the initial copyright and welcome R messages.

<infile>
e.g. your. R script: `myscript1.R`

<outfile>
If no output file name `outfile` is provided, the name of `infile` is taken as default, appending the extension `.Rout` to it. I.e. here: `myscript1.Rout`. If you open this file, you will see the welcome message that appears everytime you open a new session in R, followed by the R commands (i.e. R code) that were executed, and finally an additional command at the end of the file with information about the execution time.