Download DICE:

git clone https://bitbucket.org/vperret/dice
User guide and installation: https://bitbucket.org/vperret/dice/wiki/browse/

Install Requirements:
Cmake: sudo apt install cmake
gsl: sudo apt-get install libgsl-dev
FFTW3: sudo apt-get install -y fftw3
Sudo apt-get install fftw3-dev

Steps to build DICE:
```
$ cd dice
$ mkdir build
$ cd build
$ cmake ..
OR cmake .. -DGSL_PATH=/path/to/gsl -DFFTW3_PATH=/usr/lib/x86_64-linux-gnu/
$ make
$ make install
```

If you have a problem with the libdicetools library:
Download the library from here: libdicetools_link
Then place it in your DICE build directory (e.g., /home/rory/Downloads/dice/build/lib)

Convert Gadget file to ascii format using python code:
Original files from here: https://github.com/jveitchmichaelis/pygadgetreader
I made a Jupyter notebook version here: readgadgetfile.ipynb
Requirements: python, numpy, h5py
```
>pip install h5py # if you need h5py, this is how to install it
```

Try with the Jupyter notebook. But, if you need to install the original files, modify bash file to include:
```
PYTHONPATH=/path/to/pygadgetreader:$PYTHONPATH
export PYTHONPATH
```
Then, Install converter with:
```
>python3.8 setup.py build     ## this builds the module
>python3.8 setup.py install   ## this installs the module, may require sudo
```

Compiling Fortran programs (.f)
```
>gfortran -o prog prog.f -O3 -ffixed-line-length-none
```
'prog' is the executable that will be produced and can be run with './prog'.
'prog.f' is the ascii code file. '-O3' is optimisation level 3 for fast run speeds,
'-ffixed-line-length-none' allows any line length in the ascii code file.

Compiling the N-body tree-code gf (has a dummy mpi file)
```
>gfortran -o gf gf.f mpif.f -O3 -ffixed-line-length-none
```
'

Running gf:
gf' is the executable that will be produced and can be run with:
```
> ./gf &
```
gf requires 4 files to run:

(1) gi_par.asc # the parameters file controlling how it runs (ascii format)
(2) gr_gas0000.drt, gr_str0000.drt, gr_drk0000.drt # the initial conditions files for the gas, stars and dark matter particles respectively (binary format)

The initial conditions files can be made by converting ascii files into binary files using the ascii2drt.f program provided

Output from gf:
While running, gf will produce output files including:

1) go_log.asc # a log file, showing how time evolves as it runs
2) go_dia.asc # a diagnostics file that evolves as the simulation runs
3) Restart files in case the simulation is interrupted and needs to be restarted from a later point (e.g., gr_gas0200.drt where 200 in the number of timesteps)
4) gf_str.drt, gf_gas.drt, gf_drk.drt # the main output from gf, including particle positions and velocities in each snapshot for stars, gas and dark matter respectively. Files are binary format, and all snapshots are combined together in these files.

The main output files are binary format but can be read and converted in ascii format using the snapread.f program that is provided. You can use this to analyse all of the snapshots of the simulation while it is running or when it is finished.