1. It provides a script that includes the commands necessary to compile, link and create an executable with the gcc compiler, indicating what action is performed in each of the sentences in the script. ATTENTION, a Makefile file is not being requested

```
#!/bin/bash
--print message
echo "Compiling e1..."
--go to desired folder
cd p1_e1
--compile and link all the modules, and creates an executable “p1_e1_executable”
gcc -o p1_e1_executable p1_e1.c node.c
--return to the beginning folder
cd ..
echo "Compiling e2..."
cd p1_e2
gcc -o p1_e2_executable p1_e2.c node.c graph.c
cd ..
echo "Compiling e3..."
cd p1_e3
gcc -o p1_e3_executable p1_e3.c node.c graph.c
cd ..
```

2. Briefly justify if the following implementations of these functions are correct and if they are not, justify the reason why (suppose that the rest of the functions have been declared and implemented as in practice)

a) it is not correct because is an ADT we can not do the initialization of the node on the main, we have to call its ADT specific function.

b) it is correct because first the function is decelerated in the h document then we have the c document that implement the function and then the main in which we call the function.

c) it is not correct because the function node_ini needs a ** and in the main the only pointer that is declared is *n1.

3. Would it be possible to implement the nodes copy function using the following prototype
STATUS node_copy (Node nDest, const Node nOrigin);

No, it can´t be possible, because a function status only can return ok or error and this type of function needs to return a Node pointer, or it can be implemented as a status function but receiving Node nDest as a pointer.

4.Is the pointer Node * essential in the prototype of the function int node_print (FILE * pf, const Node * n); or it could be int node_print (FILE * pf, const Node p); ? If the answer is yes: Why? If the answer is no: Why is it used, then?

It is not essential but is better to use Node *n because we need to know where is the node that we are using we need to have that memory reference in one site and we allocate that memory in the function node_init so we need to use Node *n for refer to that information that has been allocated.

The function is implemented in that way because it takes less processing time and memory to pass a pointer to a function rather than make a copy of the whole structure and pass it to the function as well.

5.What changes should be made in the function of copying nodes if we want it to receive a node as an argument where the information should be copied? That is, how should it be implemented if instead of Node * node_copy (const Node * nOrigin), it would have been defined as STATUS node_copy (const Node * nSource, Node * nDest)?

The first thing we should do is check that the nOrigin and nDest are not null and then instead of creating a new node in the function and return this node, we should equalize the two nodes that have been given as arguments by calling to the ADT's functions and return the nDest one.

Would the following be valid: STATUS node_copy (const Node * nSource, Node ** nDest)? Discuss the differences?.

No it will not be valid because as we can see the function receives two nodes, one that is one pointer and the second one that is a double pointer and when we equalize by calling the ADT's functions we will have an error.

6.Why should the functions of Appendix 4 not be public functions?

Because in this ADT exercise the types of functions that are in Appendix 4 are part of the code that has no to be shown to the user , because this functions have important part of the c code that have to be secret for the user as we have seen on ADT's class, the user can not see the c part of the code they only can call the public functions.