1. **Procedure:**

```
def expand(grammar: dict) -> str:
    todo = random.choice(grammar["<start>"])
    result = ""

    while todo:
        production = todo.pop(0)

        if production in grammar:
            replacement = random.choice(grammar[production])
            todo = replacement + todo
        elif len(production)>0 and production[0].isalnum():
            result = result + " " + production
        else:
            result = result + production

    return result.strip()
```

Procedure Call:

```
if __name__ == "__main__":
    if len(sys.argv) > 1:
        nonterminals = parse_grammar(open(sys.argv[1]).read())
    else:
        nonterminals = poem()
    print(expand(nonterminals))
```

2. **Describe in general what the identified procedure does and how it contributes to the overall functionality of the program.**

The "expand" function creates coherent sentences by moving the nonterminal and terminal tokens from the todo list around. The program starts a while loop, and in each iteration of the loop, the procedure takes the first token from the todo list and examines it. Once the loop is complete, the program returns the result string, which is the random sentence generated.

3. **Explain in detailed steps how the algorithm implemented in the identified procedure works. (Your explanation must be detailed enough for someone else to recreate it.)**

The program enters a loop that continues as long as there are items in the "todo" list. Within each iteration of the loop, the first token from the todo list is processed. When a nonterminal is located in the grammar dictionary, the “random.choice” function expands the nonterminal randomly, and then adds it back to the todo list. If the token is terminal, it is appended to the result string. The .strip() function also removes any extra space before a comma. The result string is returned when there are no remaining terminals or nonterminals in the todo list.