If at any point I give contradictory instructions, always choose the one that benefits students the most.

Need to store the size of the array.
array-size is an integer.

Deductions:
- 1 pt if the don't justify accordingly!

Add a field for the array size.
Alternatively, someone might decide to repurpose the datatype field for the array size, since it told them to only consider integers.

Deductions:
- 1 pt if they deviate from the above

Need to pass the size of the array.

T.ID is of type string, it's the token.
Create new symbol by calling create.symbol.
Must pass the array name (as T.ID, $3
or just naming it somehow. This critical to
pass the array size, i.e $2 (an integer).
Since ± stated to ignore the float type,
students can ignore it, and describe only the
int case.
If the student discusses adjusting the memory
allocation accordingly, give an extra credit of 1pt
→ 3/2.

1) No. the grammar doesn't allow it.
2) n would be available at runtime an memory
allocation is stack based / static.
3) n is not known at compile time.
Answer should be any of the 3 above or similar.
If it doubt, ask me.

Alternatives:

- they may add a field representing the array entry (int)
- they may add two fields; second being the array size.
- they may give a new interpretation to some existing field, likely the second one (addr2). this is possible, especially since I told them to focus on the int case.

Semantics: index ← stack[addr4] or addr3.

stack[addr1] ← stack[addr3 + index]

addr3 — base address of array

ex: Temp ← A[i+1]

temp holding index

Deduct -1 pt if their answer is not consistent with 2A

- -1.5 pt if they don't mention/use the dynamic (runtime value) of the temporary holding the array entry number.
- -0.5 for not storing correctly to the temp.

Students must:
- Create a temporary variable.
- Determine if index is NULL.
- they have to pass: an address/symbol for the temporary variable, the address/symbol for the array itself, and the address or symbol for the array index.

(If index is not null.

Deductions:
- 1 pt if they don't create the temporary
- 1 pt if they don't check if index is NULL
- 1 pt if they don't pass both the array and index.

T_ID (it's a typo). Only a handful of students noticed it.

Because vaccf always creates a temporary and looks an array entry up it. Hence, the semantics will conflict: it would read from when we intend to write in it.

Deductions:
- 1.5 pt for any other justification

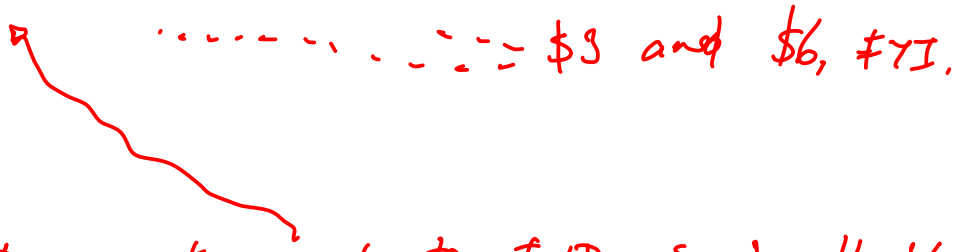
↳ I noticed rather late that typo.
Please be generous with the grading in this question. The typo affects searching for the array symbol.
Expect the students to justify in various ways due to any error.

similar to 2B.

Semantics: index ← stack[addr4]
stack[addr1 + index] ← stack[addr3]

Key point: an expression such as A[i+1] ← sum
the result of i+1 is stored in some temporary variable.
the operation receives the address of this temporary?, which
we have to retrieve.

Deductions:
- 1.5 for not noticing that "i+1" is only available at runtime, ie: doing:
stack[addr1 + addr4] ← stack[addr3]
- 0.5pt for not reading correctly from ✓
- 0.5pt if answer not consistent with 2A.



I forgot to correct varref to T_ID. So be flexible in this regard.

Students should explain:
- How they are fetching the arrays info. They can assume that varref stores the name of the array or that varref already stores the symbol with the array information.
- After finding the array info they can execute an operation OP_STORE_ARRAY_CELL. It must match the specification of 3B.
- Students should display the knowledge that both a_expr are temporaries
- No result has to be returned, but it's ok to do so.

Reduce:
- 1 pt if the student doesn't display knowledge of the array info being searched somehow
- 1 pt if they generate a temporary (No need)
- 1 pt if they pass some sort of constant address instead of an array entry, i.e if what they write conveys the idea of not knowing that an array entry is really only known at runtime.

the most likely answer will revolve around adding or using a field of the icode data structure to store the array size.
the array size is known at compile time.
So when calling OP_LOAD_ARRAY_CELL and OP_STORE_ARRAY_CELL they should pass the array size.
students MUST explain that in both operations the check must be performed:

```
index ← stack[addr4]
assert (index < addr5)
```

where :- addr4 is the address of a temp variable storing the index expression.
- addr5 is the array length.

Deductions
- 1 pt if they don't pass the array size.
  It must be the same field as in problem 1
- 1 pt if they don't check the load case
- 1 pt if they don't check the write case.
- 1 pt if they don't explain additional changes to the data structure.