4. Mark each of the following statements as valid or invalid. If a statement is invalid, explain why.

   a. `A = B;`

   b. `list->link = A->link;`

   c. `list->link->info = 45;`

   d. `*list = B;`

   e. `*A = *B;`

   f. `B = A->link->info;`

   g. `A->info = B->info;`

   h. `list = B->link->link;`

   i. `B = B->link->link->link;`

5. Write C++ statements to do the following:

   a. Make `A` point to the node containing `info` `23`.

   b. Make `list` point to the node containing `16`.

   c. Make `B` point to the last node in the list.

   d. Make `list` point to an empty list.

   e. Set the value of the node containing `25` to `35`.

   f. Create and insert the node with `info` `10` after the node pointed to by `A`.

   g. Delete the node with `info` `23`. Also, deallocate the memory occupied by this node.

6. What is the output of the following C++ code?

```
p = list;

while (p != NULL)
    cout << p->info << " ";
    p = p->link;
cout << endl;
```

7. If the following C++ code is valid, show the output. If it is invalid, explain why.

   a.
```
s = A;
p = B;
s->info = B;
p = p->link;
cout << s->info << " " << p->info << endl;
```

   b.
```
p = A;
p = p->link;
s = p;
p->link = NULL;
s = s->link;
cout << p->info << " " << s->info << endl;
```