ENGG1340 Computer Programming II

Module 8 Self-Review Exercise Solution

1. Given the following declarations:

```
int x;
int * ptr1, * ptr2;
double * ptr3;
```

Which of the following statements is invalid? Explain why.

(a) `ptr1 = ptr2;`

(b) `x = ptr1;`

(c) `*ptr3 = *ptr2;`

(d) `x = *ptr2;`

(e) `ptr1 = &ptr2;`

(f) `x = &ptr1;`

(b) cannot assign a "pointer to int" (int *) to "int"

(e) cannot assign a "pointer to a pointer to int" (int **) to "pointer to int" (int *)

(f) cannot assign a "pointer to a pointer to int" (int **) to "int"

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

```
int x;
int y;
int * p = &x;
p = &y;
*p = 10;
x = y + 20;
p = &x;
y = 25;
*p = 50;
cout << *p << " " << x << " " << y << endl;
```

```
50 50 25
```

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

```
int *x = new int;
int *y;
*x = 60;
y = x;
*y = *y + *x;
x = new int;
*x = *y - 20;
cout << *x << " " << *y << endl;
```

```
100 120
```

4. What is wrong with the following C++ code?

```
double *x = new double;
double *y = new double;
*x = 10;
y = x;
delete x;
delete y;
x = new double;
*x = 20;
cout << *x << " " << *y << endl;
```

In line 4, `y` points to the same memory location as `x`. Hence, after the memory location pointed to by `x` is deleted in line 5, the same location pointed to by `y` cannot be deleted again.

5. Given the following declarations:

```
int * aPtr;                // aPtr should point to array a
int n;
int a[5] = { 1, 2, 3, 4, 5 };
```

State the error in each of the following statements:

(a) `++aPtr;`

(b) `n = aPtr;           // use pointer to get the first value of array`

(c) `n = *aPtr[2];       // assign 2nd element of array to n`

(d)
```
// print entire array
for (int i = 0; i <= 5; ++i)
      cout << aPtr[i] << endl;
```

(e) `++a;`

(a) `aPtr` is not initialized. Initalize `aPtr` with `aPtr = a;`

(b) `aPtr` is not dereference. Change the statement to `n = *aPtr;`

(c) `aPtr[2]` is not a pointer and should not be dereferenced. Change `*aPtr[2]` to `aPtr[2]`.

(d) Referring to the 6th element of the array using `aPtr[5]` which is out of bound. Use the `<` operator instead of the `<=` operator in the loop control of the `for` statement.

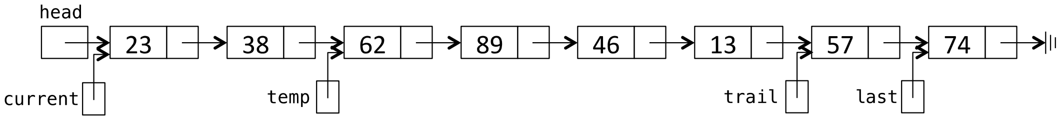
(e) `a` is an array name that cannot be modified using pointer arithmetic. Use a pointer variable instead, or subscript the array name to refer to an individual element.

6. What is stored in array after the following code executes?

```
int array[7] = { 4, 8, 9, 1, 13, 32, 20};
int * ptr = array;
*ptr = *ptr + 5;
ptr = ptr + 2;
*ptr = (*ptr) - *(ptr - 1);
ptr++;
*ptr = 5 * (*ptr) - 2;
```

```
9 8 1 3 13 32 20
```

Consider the linked list shown below:



Assume that the nodes are defined as the following structure:

```
struct Node {
   int info;
   Node * next;
};
```

and that the pointers `head`, `current`, `temp`, `trail` and `last` are all of type `Node *`.

Use the above list to answer questions 1 to 5.

7. What is the output, if any, of each of the following statements:

(a) `cout << current->info;`

(b) `cout << temp->next->next->info;`

(c) `cout << last->next->info;`

(a) `23`

(b) `46`

(c) invalid

8. What is the value of each of the following relational expression ?

(a) `current->next == temp`

(b) `trail->next->next == 0`

(c) `head == current`

(a) `false`

(b) `true`

(c) `true`

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

(a) Set the `info` of the second node to `100`.

(b) Make `trail` point to the node before `temp`.

(c) Write a `while` loop to make `current` point to the node with `info` 46.

(a) `current->next->info = 100;`

(b) `trail = current->next;`

(c)
```
while (current->info != 46)
    current = current->next;
```

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

(a) Create the node with `info` 90 and insert between `trail` and `last`.

(b) Delete the last node of the list and also deallocate the memory occupied by this node. After deleting the node, make `last` point to the last node of the list and the link of the last node must be `NULL`.

(a)

```
temp = new Node;
temp->nfo = 90;
temp->next = last;
trail->next = temp;
```

(b)

```
delete last;
trail->next = NULL;
last = trail;
```

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

```
temp = current;                  // Line 1
current = current->next;         // Line 2
current->next = last;            // Line 3
trail = current-> next;          // Line 4
trail = trail->next;             // Line 5
cout << current->info << " " << trail->info << endl;       // Line 6
```

After the execution of the statement in Line 5, `trail` is `NULL`, so `trail->info` does not exist. This code will result in a run-time error.