CSCI 201: Fundamentals of Computer Science
(Instructor: Pranava K. Jha)

Solution to exercises on pointers and dynamic arrays

1. Consider the following declarations:

```c
int i = 5, j = 7, *p = &i, *q = &j;
float x = 2.0, y = 8.0, *r = &x, *s = &y;
```

Mark the following statements as valid or invalid. If a statement is valid, then state which variable changes, and if it is invalid, then state the reason why it is invalid.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Valid or invalid?</th>
<th>If valid, which variable is changed?</th>
<th>If invalid, why?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. i = j;</td>
<td>valid</td>
<td>i</td>
<td></td>
</tr>
<tr>
<td>2. i = i - x;</td>
<td>valid</td>
<td>i</td>
<td></td>
</tr>
<tr>
<td>3. i = *r;</td>
<td>valid</td>
<td>i</td>
<td></td>
</tr>
<tr>
<td>4. p = &amp;j;</td>
<td>valid</td>
<td>p</td>
<td></td>
</tr>
<tr>
<td>5. p = q;</td>
<td>valid</td>
<td>p</td>
<td></td>
</tr>
<tr>
<td>6. *p = j;</td>
<td>valid</td>
<td>i</td>
<td></td>
</tr>
<tr>
<td>7. *p = *s;</td>
<td>valid</td>
<td>i</td>
<td></td>
</tr>
<tr>
<td>8. p = j;</td>
<td>invalid</td>
<td></td>
<td>Cannot place an integer value in a pointer.</td>
</tr>
<tr>
<td>9. p = *j;</td>
<td>invalid</td>
<td></td>
<td>Cannot de-reference an integer variable.</td>
</tr>
<tr>
<td>10. p = s;</td>
<td>invalid</td>
<td></td>
<td>Cannot place a float address in an integer pointer.</td>
</tr>
<tr>
<td>11. p = *q;</td>
<td>invalid</td>
<td></td>
<td>Cannot place an integer value in a pointer.</td>
</tr>
<tr>
<td>12. p = &amp;x;</td>
<td>invalid</td>
<td></td>
<td>Cannot place a float address in an integer pointer.</td>
</tr>
<tr>
<td>13. *p = s;</td>
<td>invalid</td>
<td></td>
<td>Cannot place a pointer in a non-pointer variable i.</td>
</tr>
<tr>
<td>14. *p = &amp;i;</td>
<td>invalid</td>
<td></td>
<td>Cannot place an address in a non-pointer variable.</td>
</tr>
</tbody>
</table>
2. Consider the following declarations:

```c
int i[5], *p1 = &i[0];
float x[5], *p2 = &x[0];
```

Mark the following statements as valid or invalid. If a statement is valid, then state which variable changes, and if it is invalid, then state the reason why it is invalid.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Valid or invalid?</th>
<th>If valid, which variable changes?</th>
<th>If invalid, why?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ( x = x + 1 );</td>
<td>invalid</td>
<td>—</td>
<td>Cannot change the address of an array.</td>
</tr>
<tr>
<td>2. ( x = p2 + 1 );</td>
<td>invalid</td>
<td>—</td>
<td>Adding 1 to a pointer is okay, but the address of an array cannot be changed.</td>
</tr>
<tr>
<td>3. ( p2 = x + 1 );</td>
<td>valid</td>
<td>( p2 )</td>
<td>—</td>
</tr>
<tr>
<td>4. ( p2 = p2 + 1 );</td>
<td>valid</td>
<td>( p2 )</td>
<td>—</td>
</tr>
</tbody>
</table>

3. Present the output in its content and form when the following program is executed.

```c
#include <iostream>
using namespace std;

int main()
{
    double x, y, z, temp;
    double *p, *p1, *p2, *p3;
    x = 3.4; y = 5.6; z = 7.8; p1 = &x; p2 = &y; p3 = &z;
    temp = x; x = y; y = z; z = temp; p = p1; p1 = p2; p2 = p3; p3 = p;
    cout << x << ' ' << y << ' ' << z << endl;
    cout << *p1 << ' ' << *p2 << ' ' << *p3 << endl;
    return 0;
}
```

```
5 . 6
7 . 8
3 . 4
```

```
7 . 8
3 . 4
5 . 6
```
4. Given the declaration

```c
int x;
int* p;
int* q;
```

mark the following statements as valid or invalid. If a statement is invalid, explain why.

- `p = q;`
- `*p = 56;`
- `p = x;`
- `*p = *q;`
- `q = &x;`
- `*p = q;`

5. Present the output of the following piece of code.

```c
int x;
int y;
int *p = &x;
int *q = &y;
*p = 35;
*q = 98;
*p = *q;
cout << x << " " << y << endl;
cout << *p << " " << *q << endl;
```

Output:

```
98 98
98 98
```

6. Present the output of the following piece of code.

```c
int x;
int y;
int *p = &x;
int *q = &y;
x = 35; y = 46;
p = q;
*p = 78;
cout << x << " " << y << endl;
cout << *p << " " << *q << endl;
```

Output:

```
35 78
78 78
```
7. Present the output of the following piece of code.

```c++
int *p;
int *q;
p = new int;
q = p;
*p = 46;
*q = 39;
cout << *p << " " << *q << endl;
```

```
39 39
```

8. Present the output of the following piece of code.

```c++
int *p;
int *q;
p = new int;
*p = 43;
q = p;
*q = 52;
p = new int;
*p = 78;
q = new int;
*q = *p;
cout << *p << " " << *q << endl;
```

```
78 78
```

9. Present the output of the following piece of code.

```c++
int x; int *p; int *q;
p = new int[10];
q = p; *p = 4;
for (int j = 0; j < 10; j++)
{
    x = *p;
p++;
    *p = x + j;
}
for (int k = 0; k < 10; k++)
{
    cout << *q << " ";
    q++;
}
cout << endl;
```

```
4 4 5 7 10 14 19 25 32 40
```
10. Present the output of the following piece of code.

```c++
int *p;
int *q;
int j;
p = new int[5];
p[0] = 5;
for (j = 1; j < 5; j++)
    p[j] = p[j-1] + 2*j;
cout << "Array p: ";
for (j = 0; j < 5; j++)
    cout << p[j] << " ";
cout << endl;
q = new int[5];
for (j = 0; j < 5; j++)
    q[j] = p[4-j];
cout << "Array q: ";
for (j = 0; j < 5; j++)
    cout << q[j] << " ";
cout << endl;
```

Array p: 5 7 11 17 25
Array q: 25 17 11 7 5