1. The output of the program is: 1 14 3

2. (a) 2D static arrays (e.g., arr2D) are always stored in memory as 1D arrays (e.g., arr1D). To find where element arr2D[i][j] is stored in arr1D, the following operation needs to be performed:

\[ \text{arr2D}[i][j] \rightarrow \text{arr1D}[i*\text{noCols}+j] \]

Therefore, the number of columns is required to establish the correspondence between arr2D and arra1D (see slide # 45 from C++ Review)

(b) The C++ statements for dynamically allocate an array A with 3 rows and 5 columns are given below:

```cpp
int **arr2D, i;
arr2D = new int* [3];
for(i=0; i<3; i++)
    arr2D[i] = new int [5];
```

(c) This is similar to the diagram on slide #47 from the C++ review lecture

(d) To find the location of element arr2D[i][j], first it find the base address of row i which is stored in arr2D[i]. Then, using j as an offset from the base address, it finds the address of arr2D[i][j].

(e) The C++ statements for dynamically deallocate an array A with 10 rows and 5 columns are given below:

```cpp
for(i=0; i<10; i++)
    delete [] arr2D[i];
delete [] arr2D;
```
3. Exercise 18 (page 188)
   (a) Yes, (b) No, (c) No, (d) Yes, (e) No, (f) Yes, (g) Yes, (h) No
   (i) No, (j) Yes, (k) Yes, (l) No, (m) No, (n) Yes, (o) No

4. Exercise 19 (page 188)
   #include <iostream>

   int main()
   {
      int* ptr;
      int* temp;
      int x;

      ptr = new int;
      *ptr = 4;
      temp = new int; // temp needs to be initialized prior to assignment
      *temp = *ptr;
      cout<<*ptr<<*temp;
      x = 9;
      *temp = x;
      cout<<*ptr<<*temp;
      ptr = new int;
      *ptr = 5; // CHANGED
      *temp = *ptr;
      cout<<*ptr<<*temp;
   }

   Without spacing information, the output is 444959