1.
   a. ```c++
      void func() {
        int sum = 0;
        int aBag[4] = {1, 2, 3, 4};

        for (int i = 0; i < sizeof(aBag)/sizeof(aBag[0]); i++) {
          sum += aBag[i];
        }
        return sum;
      }

      ```

   b. ```c++
      bool replace(item src, item dst, bag *b) {
           for (int i = 0; i < sizeof(b)/sizeof(b[0]); i++) {
          if (b[i] == src) {
            b[i] = dst;
            return true;
          }
        }
        return false;
      }
      ```

2. ```c++
   #include <iostream>
   #include <stdio.h>

   class Rectangle {
   private:
     int width;
     int height;
   public:
      Rectangle(int w, int h) : width(w), height(h) {}

     void setWidth(int w) { width = w; }
     void setHeight(int h) { height = h; }
     int getWidth() { return width; }
     int getHeight() { return height; }
     int getArea() { return width*height; }
     int getPerimeter() { return width+height; }
   };

   int main() {
     Rectangle rect(10, 20);
     printf("w: %d, h: %d\n", rect.getWidth(), rect.getHeight());

     rect.setWidth(20);
     rect.setHeight(40);

```
    printf("w: %d, h: %d\n", rect.getWidth(), rect.getHeight());
    printf("area: %d, perimeter: %d\n", rect.getArea(),
rect.getPerimeter());

    return 0;
  }
  ```
```

---

3.

1. ```c++
   #include <iostream>

   using namespace std;

   class Node {
   public:
     string name;
     Node *prev;
     Node *next;

     Node(string n) : name(n) {
       prev = NULL;
       next = NULL;
     }
   };

   /*
   1. 將新增的 node 作為原本 head node 的 prev node
   2. 將新增的 node 的下個 node 設為現在的 head node
   3. 移動 headPtr 至該 node，作為新的 head node
   */
   void addNodeBegin(Node **hptr, Node *n) {
     (*hptr)->prev = n;
     n->next = *hptr;
     *hptr = n;
   }
   /*
   要先確定是否有下一個 node，而 Figure 的情況是有下個 node，所以接下來都會以有下個 node 作為前提。
   1. 將 head node 記錄起來 (在這邊為 tmp)，避免待會在刪除 node 時找不到 node
   2. 把下個 node 的 prev 設為 NULL，代表沒有上個 node 已經被移除
   3. 移動 headPtr 至下一個 node
   4. 刪除 Step 1 中記錄的 node (tmp)，這樣做能避免在 Step 3 時已經移動 headPtr 而找不到要被刪除的 node 的情況
   */
   void removeNodeBegin(Node **hptr) {
     if ((*hptr)->next != NULL) {
       Node *tmp = *hptr;
```

```
    (*hptr)->next->prev = NULL;
    *hptr = (*hptr)->next;
    delete(tmp);
  } else {
    delete(*hptr);
    *hptr = NULL;
  }
}

void showAll(Node *h) {
  while(h != NULL) {
           cout << h->name << ' ';
           h = h->next;
     }
  cout << '\n';
}

int main() {
  Node *a = new Node("Able");
  Node *b = new Node("Baker");
  Node *c = new Node("Jones");
  Node *headPtr = a;

  addNodeBegin(&headPtr, b);
  addNodeBegin(&headPtr, c);
  showAll(headPtr);

  removeNodeBegin(&headPtr);
  showAll(headPtr);

  return 0;
}
```