SOLUTIONS

```
void insertOrdered(DoublyNode* newNode,
                   DoublyNode* current)
{
   // CODE BLOCK I
   if ((current == trailer) ||
       (newNode->score <= current->score))
   {
      newNode->next = current;
      newNode->prev = current->prev;
      current->prev->next = newNode;
      current->prev = newNode;
   }
   else
      insertOrdered(newNode, current->next);

   // CODE BLOCK II
   if ((current->next == trailer) ||
       (newNode->score <= current->next->score))
   {
      newNode->next = current->next;
      newNode->prev = current;
      current->next->prev = newNode;
      current->next = newNode;
   }
   else
      insertOrdered(newNode, current->next);

   // CODE BLOCK III
   if((newNode->score >= current->next->score) &&
       (current->next != trailer))
      insertOrdered(newNode, current->next);
   else
   {
      newNode->next = current->next;
      newNode->prev = current;
      current->next->prev = newNode;
      current->next = newNode;
   }

   // CODE BLOCK IV
   if ((newNode->score >= current->score) &&
       (current != trailer))
      insertOrdered(newNode, current->next);
   else
   {
      newNode->next = current;
      newNode->prev = current->prev;
      current->prev->next = newNode;
      current->prev = newNode;
   }
}

int main()
{
   DoublyLinkedList list;  DoublyNode* newNode;
   newNode          = new DoublyNode;
   newNode->elem    = "Paul";
   newNode->score   = 720;
   list.insertOrdered(newNode, list.header);
}
```

1. Which code blocks insert Paul without any recursive call? **(25P)**

Assume that **Header**'s and **Trailer**'s scores are **0**.

*You'll loose **5P** from wrong answer.*

(A) I, II

(B) I, III

(C) II, III

(D) II, IV

(E) III, IV

2. Considering **Code Block II**, how many times does the `insertOrdered()` function call itself recursively when adding nodes that have scores 720, 590, 660 and 1105 respectively? **(25P)**

Assume that **Header**'s and **Trailer**'s scores are **0**.

4

```
int binarySum(int A[], int i, int n)
{
   if (n == 1)
      return A[i];
   else
   {
      int Sum = binarySum(A, i, n / 2) +
                binarySum(A, i + n / 2, n / 2);
      cout << Sum << " ";
      return Sum;
   }
}

int main()
{
   int A[8] = {2,4,6,8,10,12,14,16};
   int binSum = binarySum(A, 0, 8);
}
```

3. What is the output of the program above? **(25P)**

6	14	20	22	30	52	72

4. How many times does the **binarySum()** function call itself recursively? **(25P)**

14