4) 16 pts

You are given an unsorted array $A[1..n]$ of positive integers. You want to maximize the number of points you get by performing the following operations.
Pick any $A[i]$ and delete it to earn $A[i]$ points. Afterwards, you must delete elements $A[i-1]$ and $A[i+1]$ (if they exist) for which we won't get any points. This process will be called one move. Develop an efficient dynamic programming solution to return the maximum number of points you can earn with exactly k moves, where $k<n$.

Example: A=[1,8,2], k=1 Solution: 8
Example: A=[1,8,2], k=2 Solution: 3 (if we pick 8 we won't be able to make 2 moves because all elements will be deleted with that first move)

Note: if the array is of size 1 or 2 we cannot make more than one move. If the array is of size 3 or 4 we cannot make more than 2 moves, etc.

I. Define (in plain English) the subproblems to be solved. (3 pts)

Solution 1	*OPT(i,j)* = the maximum number of points achievable when the input array is the subarray *A[1..i]* and the number of moves to make is *j*.
Solution 2	*OPT(i,j,k)* = the maximum number of points achievable when the input array is the subarray *A[i..j]* and the number of moves to make is exactly *k*.
Solution 3	*OPT(i,j)* = the maximum number of points achievable when the input array is the subarray *A[1..i]* and the number of moves to make is exactly *j*, with the *j*-th move deletes element *i* for points.

Rubric:

- (-3pts)
  - The student did not give an answer for this part.
  - The definition of the subproblem is not correct, that is not satisfying the optimal-substructure property.
    - Not considering the constraint "exactly k moves":
      - OPT(i,j) = maximum number of points for A[i,j]
      - OPT(i) = maximum number of points for A[i]
    - Not a valid definition of subproblems to achieve efficient algorithm
      - OPT(R,k) = maximum number of points for the array R with exactly k moves, where R can be any subsequence of A after deleting some elements. There may be O(2^n) subsequences R and Q4 requires an iteration-based solution.
  - Does not define an actual subproblem:
    - "Choose *A[n]* or do not choose *A[n]"*
- (-1pts)
  - The definition of the subproblem is not clear.
    - OPT(i,j) = maximum number of points for A[1..i], while j is not mentioned.
    - Let *OPT(i,k)* be the maximum number of points you can earn with *k* moves and number *i*.
- (-0pts)
  - Correct Answer.

II. Write a recurrence relation for the subproblems (5 pts)

Solution 1	$OPT(i,j) = max(OPT(i-1,j), A[i]+OPT(i-2,j-1))$
Solution 2	$OPT(i,j,k) = max\{ OPT(i,u-2,v) + OPT(u+2,j,k-1-v) + A[u] \}$ *for any* $i \le u \le j$ *and* $0 \le v \le k-1$.
Solution 3	$OPT(i,j) = max\{ OPT(u,j-1) \} + A[i]$ *for any* $1 \le u \le i-2$.
Solution 4	$OPT(i,j) = max_{0 \le p \le i} \{ OPT(p-2, j-1) + A[p] \}$

Rubric:

- (-5pts) If the previous question is not answered, or, the recurrence relation is not correct, or no answer. There are some examples:
  - Missing Loop for Solution 2: OPT(i,j,k) = max{ OPT(i,u-2,k-1) + OPT(u+2,j,k-1) + A[u] } for any i ≤ u ≤ j. This recurrence misses loop and also the coefficient "k-1" for OPT(i,u-2,k-1) or OPT(u+2,j,k-1) is wrong.
  - OPT(i,j) = max{ OPT(i-2,j-1), OPT(i+2,j-1) } + A[i].
- (-2pts) Per major error.
  - If the coefficient/loop condition is not correct, but the formula (and the definition of subproblems) is almost correct.
    1. Coefficient mistakes:
       a. OPT(i,j) = max(OPT(i-1,j), A[i]+OPT(i-3,j-1)) where the coefficient 3 is wrong
       b. OPT(i,j) = max(OPT(i-1,j), A[i-1]+OPT(i-3,j-1)) where the coefficient i-1 is wrong
    2. Loop condition mistakes:
       a. OPT(i,j) = max(OPT(i-2,j-1), OPT(i-3,j-1)) + A[i]. Actually this can be categorized as a bad attempt to Solution 1 as well.
       b. OPT(i,k) = max(OPT(u,j-1)) + A[u] for $1 \le u \le i-3$ where the loop condition is wrong.
  - Writes two cases, but does not combine to a single equation:
    Pick $A[i]$: $OPT(i,j) = OPT(i-2,j-1) + A[i]$
    Do not pick $A[i]$: $OPT(i,j) = OPT(i-1,j)$
  - The correct recurrence relation is a component of the student's answer, but the student's answer is not correct.
    $OPT(i,k) = max_{k<n} \{ OPT(i-1,k), OPT(i-2,k-1) + A[i] \}$
  - Mixed up pluses and minuses:
    Subproblem is defined $OPT(i,j)$ = max points for array $A[1..i]$ and j moves, but recurrence relation is defined $OPT(i,j) = max\{ OPT(i+1,j), OPT(i+2,j-1) + A[i] \}$
  - Mixes up i and j in some parts:
    $OPT(i,j) = max \{ OPT(i-1, j), OPT(i-2, j-1) + A[j] \}$
- (-0pts) Incorrectly uses both i and n together in the recurrence relation.
  - $OPT(n,k) = max(OPT(n-1,k), OPT(n-2, k-1) + A[i])$

        - *OPT(n,k) = max(OPT(i-1,k), OPT(i-2, k-1) + A[i])*
    - (-0pts) Using *n* or *k* in the recurrence relation for this part. However, may take points off in question III-iii for using *n* or *k* as the variable names in the iteration for-loops.
        - *OPT(i,k) = max(OPT(i-1,k), OPT(i-2,k-1) + A[i])*
    - (-0pts) Per typo. "Typos" are errors that are small enough that they don't suggest a misunderstanding on the part of the student of how to correctly solve the problem.
        - Referring to the array *A* as *W*.
    - (-0pts) Correct Answer.

III. Using the recurrence formula in part b, write pseudocode using iteration to compute the maximum number of points that can be earned. (6 pts total)
Make sure you specify:

i. the base cases and their values (2 pts)

Solution 1	• -Initialize $OPT(i,0) = 0$ for all integers $i \geq 0$, i.e., the maximum score achievable with exactly zero moves is always zero. • -Initialize $OPT(0,j) = -\infty$ for all integers $j \geq 1$, i.e., it is infeasible to make $j \geq 1$ moves given an array A with length 0. • -Initialize $OPT(1,1) = A[1]$. • -Initialize $OPT(1,j) = -\infty$ for all integers $j \geq 2$, i.e., it is infeasible to make $j \geq 2$ moves given an array A with length 1.
Solution 2	• -Initialize $OPT(i,j,0) = 0$ for all $-1 \leq i,j \leq n+2$, i.e., the maximum score achievable with exactly zero moves is always zero. • -Initialize $OPT(i,j,k) = -\infty$ for all integers i,j,k such that $-1 \leq i,j \leq n+2$ and $i > j$ and $k \geq 1$, i.e., it is infeasible to make $k \geq 1$ moves given an array with length 0. • -Initialize $OPT(i,i,1) = A[i]$ for all integers $i \geq 1$. • -Initialize $OPT(i,i,k) = -\infty$ for all integers $i \geq 1$ and $k \geq 2$, i.e., it is infeasible to make $k \geq 2$ moves given an array with length 1. The second and forth conditions are critical to ensure "exactly k moves". Neal: the third and fourth base cases are actually unnecessary given the first and second base cases.
Solution 3	• Initialize $OPT(i, 0) = 0$ for all $i \geq 1$ • Initialize $OPT(i,1) = A[i]$ for all $i \geq 1$ *Note: we assume that the maximum of an empty set is defined to be $-\infty$.*

Rubric:

- (-2pts) If the previous questions are not answered correctly (that is, the student has already scored 0pts for both subquestion I and II),
- (-2pts) No answer.
- (-2pts) Insufficient base cases.
- (-1pts) If one pair of the base cases and their values is incorrect.
  - Example 1: OPT(i,0) = 0 or OPT(0,i) = 0 is not mentioned for Solution 1.
  - Example 2: OPT(i,i-1,k) = $-\infty$ or OPT(i,i-2,k) = $-\infty$ for k>0 is not mentioned for Solution 2. These two conditions are critical to Solution 2 as they are ensuring the "exactly k moves" constraint.
- (-0pts) Correct Answer for all situations. Note that, since the problem didn't mention the correct output when $k>(n+1)/2$, the $-\infty$ initializations are not taken

into consideration. Neal: The -∞ initializations are actually necessary for the algorithm to output the correct answer for nominal cases.

ii. where the final answer can be found (e.g. *OPT(n)*, or *OPT(0,n)*, etc.) (1 pt)

Solution 1	*OPT(n,k)* or max_i *{ OPT(i,k) }*
Solution 2	*OPT(1,n,k)*
Solution 3	*Max{ OPT(i,k) } for* $1 \leq i \leq n$

Rubric:

- (-1pt) If the previous questions are not answered correctly (that is, the student has already earned 0pts for either subquestion I and II),
- (-1pt) No answer
- (-1pt) Incorrect answer.
  - max OPT[i][k] for k in 1..n for Solution 1.
  - OPT(1,n)
  - OPT(n)
  - OPT(n,k) for Solution 3.
- (-0pts) Correct Answer.

iii. Rest of pseudocode (3 pt)

Solution 1	Initialization *OPT* as above. for *i=2* to *n* for *j=1* to *k* *OPT(i,j) = max(OPT(i-1,j),A[i]+OPT(i-2,j-1))* endfor endfor return *OPT(n,k)*
Solution 2	Initialize *OPT* as above. for *d=2* to *n*: for *i=1* to *n-d*: *j = i + d* for *t=1* to *k*: $OPT(i,j,t) = max_{i \le u \le j,\ 0 \le v \le t-1} \{ OPT(i,u-2,v) + OPT(u+2,j,t-1-v) + A[u] \}$ endfor endfor endfor
Solution 3	Initialize *OPT* as above. for *i=1* to *n* for *j=2* to *k* $OPT(i,j) = max_{i \le u \le i-2}\{OPT(u,j-1)\} + A[i]$ endfor endfor

Rubric:

- (-3pts) If the previous questions are not answered correctly (that is, 0 pts for either subquestion 1 and 2), or no answer, or answer is not correct.
- (-1pts) For each minor error. Don't take off more than one point for multiple errors of the same type. Some implementation details of the algorithm are not correct, with the main algorithm and the recurrence formula being correct. Note that, the wrong coefficients such as OPT(i-3,k) for solution 1, is regarded as this case.
  - Overwriting the name of either input variable *n* or *k*. For example, the name for the iterator variable in the inner loop should not be *k*:

```
for i=1 to n
        for k=1 to k
                OPT(i,k) = max(OPT(i-1,k), OPT(i-2,k-1) + A[i])
        endfor
endfor
```

- Incorrect loop construction:

```
for i=1 to j:
        for j=1 to n:
                ...
        endfor
Endfor
```

- Extra *if* statements that, if deleted, would make the pseudocode correct. Initialization *OPT* as above.

```
for i=2 to n
   for j=1 to k
      if i > 2:
         OPT(i,j) = max(OPT(i-1,j),A[i]+OPT(i-2,j-1))
      if i ==2:
         OPT(i, j-1)
      if i == 1:
        OPT(i,j) = 0
   endfor
endfor
return OPT(n,k)
```

- (-0pts) Incorrect iteration starting or ending points, if points have not already been taken off for incorrect base cases in question III-i.
- (-0pts) Correct Answer.

IV. What is the run time complexity of your solution? (2 pts)

Solution 1	Solution 2	Solution 3
$O(kn)$ or $O(n^2)$	$O(n^3k^2)$ or $O(n^5)$	$O(kn^2)$ or $O(n^3)$

Rubric:

- (-2pts) No answer for the previous question, or incorrect answer for Part II, or incorrect answer (such as $O(n)$).
- (-0pts) Correct Answer.