1) a. The while loop will be executed a maximum of $\log_2 n$ +1 times. The tight upper bound is given by O(g(n)) = f(n) , where there exists a constant c > 0 and $n_0 > 0$ such that 0 ≤ f(n) < cg(n) for all n $\geq n_0$.

In this case the f(n) = $\log_2 n$ for all n > 0

The tightest upper bound on no of times the while loop will execute is $O(\log_2 n)$.

b. The approximate complexity of power-iterative is O($\log_2 n$) .

2)

Approximate analysis:

Step #	Complexity stated as O(_)
1	O(1)
2	O(1)
4	Complexity of # of executions: O(n1) = O(n)
5	O(1)
Loop 4-5	Complexity of entire loop: O(n1) = O(n)
6	Complexity of # of executions: O(n2) = O(n)
7	O(1)
Loop 6-7	Complexity of entire loop: O(n2) = O(n)
8	O(1)
9	O(1)
10	O(1)
11	O(1)
12	Complexity of # of executions: O(r-p +1) = O(n)
13	O(1)
14	O(1)
15	O(1)
16	O(1)
17	O(1)
13-17	Complexity of single execution of loop body: O(1)
12-17	Complexity of entire loop: O(r-p+1) = O(n)
1-17	Complexity of algorithm: max(O(n1), O(n2), O(r-p+1)) = O(n)

Detailed analysis:

Step #	Cost of single execution	# of times executed
1	C1: 5	1
2	C2: 4	1
4	C4: 1 , it depends	n1 +1
5	C5: 9	n1
6	C6: 1 , it depends	n2 + 1
7	C7: 8	n2
8	C8: 4	1
9	C9: 4	1
10	C10: 1	1
11	C11: 1	1
12	C12: 1 , it depends	r-p+2 = n+1
13	C13: 8	r-p+1 = n
14	C14: 6	T1 = no of times condition true
15	C15: 3	T1
16	C16: 6	n-T1 = no of times else part is implemented
17	C17: 3	n- T1

T(n) = 5+4+n1+1+9n1+n2+1+8n2+4+4+1+1+n+1+8n+6T1+3T1+6(n-T1)+3(n-T1) , n1=n2=n/2
= 27.5n + 22
= cn + c1 where c=27.5, c1= 22

T(n) = cn+c1

3)

Level #	# of recursive executions at this level as a function of level #	Input size to each execution	Additional work done by each execution	Total work done at this level as a function of level #
0	$2^0 = 1$	n - 0 = n	1	$2^0 = 1$
1	$2^1 = 2$	n-1	1	$2^1 = 2$
2	$2^2 = 4$	n-2	1	$2^2 = 4$
n-1	2^{n-1}	n - (n-1) = 1	1	2^{n-1}
n	2^n	n-n = 0	1	2^n

$$T(n) = \sum_{i=0}^{n} 2^i = \frac{(2^{n+1} - 1)}{2-1} = 2 * 2^n - 1$$

Complexity Order = $\Theta(2^n)$

4. Let the Modulo-exponent function be represented as M(b,n,m). Here our b = 2 , n = 5, m =6. We have the recursion tree with the output as follows

M(2,5,6) <=> Return 2

$\updownarrow$

M(2,2,6) <=> Return 4

$\updownarrow$

M(2,1,6) <=> Return 2

$\updownarrow$

M(2,0,6) <=> Return 1

The table will be as follows.

Level	Input	Output
0	M(2,5,6)	2
1	M(2,2,6)	4
2	M(2,1,6)	2
3	M(2,0,6)	1

5.

Line #	Step	Single execution cost	# times executed
1	sum = max = 0	2	1
2	for i = p to q	1	q-p+2 = n+1
3	sum = 0	1	q-p+1 = n
4	for j = i to q	1	$\sum_{i=1}^{n} n - i + 2 = n^2 - \frac{n(n+1)}{2} + 2n = \boldsymbol{\frac{n^2}{2} + \frac{3n}{2}}$
5	sum = sum + A[j]	6	$\sum_{i=1}^{n} n - i + 1 = n^2 - \frac{n(n+1)}{2} + n = \boldsymbol{\frac{n^2}{2} + \frac{n}{2}}$
6	if sum > max then	3	$\boldsymbol{\frac{n^2}{2} + \frac{n}{2}}$
7	max = sum	2	$\boldsymbol{\frac{n^2}{2} + \frac{n}{2}}$
8	return max	2	1

T(n) = $2 + n+1 + n + \frac{n^2}{2} + \frac{3n}{2} + 6(\frac{n^2}{2} + \frac{n}{2}) + 3(\frac{n^2}{2} + \frac{n}{2}) + 2(\frac{n^2}{2} + \frac{n}{2}) + 2 = 6n^2 + 8n + 5$

Hence **T(n) = $\boldsymbol{6n^2 + 8n + 5}$**

6.

You must state costs in terms of n with numerical coefficients, and not using a complexity order notation, to get credit. You may assume that the for loops on lines 9-12 and 13-16 are executed n/2 times.
Which statements are executed when the input is a base case (provide line #s)? _1,2,3,4____
What is the total cost of these? ___________12______
Which statements are executed when the input is not a base case (provide line #s)? __1, 5-23___
What is the total cost of these?
________________________________2T(n/2) + 4n + 27_____________________________________
Provide the complete and precise two recurrence relations characterizing the complexity of MSS Algorithm-2:
T(n) = ______12__________ when n=1
T(n) = ____________________2T(n/2) + 4n + 27____________________________________ when n>1
Now simplify the recurrence relations by:
1. If your recurrence relation for the non-base case input has multiple terms in it besides the term representing the recursive calls, keep only the largest n-term from them and drop the others; if your recurrence relation for the non-base case input has only one other term besides the term representing the recursive calls, keep it.
2. Take the largest numerical coefficient of all terms (excluding the term representing the recursive calls) in your two recurrence relations, round it up to the next integer if it is not an integer, and replace the numerical coefficients of all other terms (excluding the term representing the recursive calls) with this coefficient.
Provide the simplified recurrence relations below.
T(n) = _________12_______ when n=1
T(n) = _______________2T(n/2) + 12n ____ when n>1

Level #	# of recursive executions at this level as a function of level #	Input size to each execution	Additional work done by each execution	Total work done at this level as a function of level #
0	$2^0 = 1$	$n/2^0 = n$	$12(n/2^0) = 12n$	$2^0 * 12(n/2^0) = 12n$
1	$2^1 = 2$	$n/2^1 = n/2$	$12(n/2^1) = 12 n/2$	$2^1 * 12(n/2^1) = 12 n$
2	$2^2 = 4$	$n/2^2 = n/4$	$12(n/2^2) = 12n/4$	$2^2 * 12(n/2^2) = 12n$
$\log n - 1$	$2^{\log n - 1} = n/2$	$n/2^{\log n - 1} = 2$	$12(n/2^{\log n - 1}) = 24$	$2^{\log n - 1} * 12(n/2^{\log n - 1}) = 12n$
$\log n$	$2^{\log n} = n$	$n/2^{\log n} = 1$	$12(n/2^{\log n}) = 12$	$2^{\log n} * 12(n/2^{\log n}) = 12n$

7.

```
MSS_Algorithm2: (A is an array(p…q) of integer)
sum,max: integer   result[1..2]: array of integer
sum=max=result[1]=result[2]=0
for i=p to q
  sum=0
  for j= i to q
     sum=sum+A[j]
     If sum>max then
        Max=sum
        result[1]=i
        result[2]=j
return result
```

8.

We guess that the solution is T(n) = O(n^2)

For this to be correct we need to prove that T(n) $\le$ cn^2 where n $\ge n_0$

Base Case:

The base case for this will be n =1. We have T(1) = 1 and for T(n) $\le$ cn^2 we should have c $\ge$ 1 and n_0 =1.

Hypothesis:

Let us assume that the condition holds true for n/2 i.e T(n/2) $\le$ c$(n/2)^2$ when n $\ge n_0$

Induction:

We need to prove that T(n) $\le$ cn^2 where n $\ge n_0$

T(n) = 3T(n/2) + n $\le$ cn^2

From hypothesis we have

T(n) = 3T(n/2) + n $\le$ 3(c$(n/2)^2$) +n

We need to fulfil the condition

3(c$(n/2)^2$) +n $\le$ cn^2

We have

$\frac{3cn^2}{4}$ + n $\le$ cn^2

cn $\ge$ 4

For the above to hold true we should have c $\ge$ 4

So for T(n) $\le$ cn^2 to be true we should have c $\ge$ 4 and n $\ge n_0$=1

Thus T(n) $\le$ cn^2 when c $\ge$ 4 and n $\ge n_0$=1