# HW1 Shir Matathias

## Question A

1) **Base Case :** $a_0 = 10$
when n=0, $a_0 = 10*5^0 = 10*1 = 10$
thus $a_0 = 10 = a_n*5^n$
**Induction Assumption :** we assume $a_n = 10*5^n$
**Induction Step :** we will show this holds for n+1.
$a_{n+1} = 10*5^{n+1}$
and by lows of exponents we can represent $a_{n+1}=10*5^n*5$
based on the induction assumption $a_{n+1} = a_n*5$ , and by that the claim holds for n+1.

2) **Base Case :** n=1
$1*2^1=1*2 = 2$
$(1-1)*2^{1+1} + 2 = 0+2 = 2$
thus the claim holds for n=1.
**Induction Assumption :** we assume $\sum_{i=1}^{n} i \cdot 2^i = (n-1) \cdot 2^{n+1} + 2$ holds $\forall n \geq 1$.
**Induction Step:** $\sum_{i=1}^{n+1} i \cdot 2^i = (n-1) \cdot 2^{n+1} + 2 + [(n-1) * 2^{n+1}] =$
$2 + 2^{n+1}(n-1+n+1) = 2^{n+1} * 2n + 2 =$
$2^{n+1} * 2^1 * n + 2 = 2^{n+2} * n + 2 = (n+1-1) * 2^{n+1+1} + 2$

3)**Base Case :** n=1 can be written as $1=2^0$.
$[1] = 2^{1-1} = 2^0 = 1$.
**Induction Assumption :** Assume $m \epsilon Z^+$ and $1 \leq m \leq k$ and that our prop. holds for n.
**Induction Step :** we divide into cases :
-k+1 is even : then $\frac{k+1}{2} \epsilon Z^+$ as $1 \leq \frac{k+1}{2} \leq k$ we know by the induction assumption that $\frac{k+1}{2}$ is the sum of distinct powers of 2.
now, we multiply by 2 and we obtain that $\frac{k+1}{2} * 2 = k+1$.
Since, each distinct power of 2 in the sum $\frac{k+1}{2}$ is multiplied by 2 ($2^1$), so each power is increased by 1 and thus the powers of 2 remain distinct.
-k+1 is odd : thus k is even. By that we know by the induction assumption that k can be expressed as a sum of distinct powers of 2.
As k is even we know $2^0=1$ does not exists (every power of 2 is even but $2^0=1$ is odd and thus it can not be in the representaion of k).
Now, we still can have the sum of k+1 to be represented as distinct powers of 2.

## Question B

1) The claim is **false** - the proof is incorrect because $\exists a,n \epsilon Z^+$ s.t. $a^n \neq 1$. for example : let $a,n = 2 \epsilon Z^+$, we know $2^2 = 4 \neq 1$ contradiction!
what went wrong ? In the base case we define n=0 and we know that $\forall a \epsilon Z^+$ we have $a^n=1$.
but in the base case we defined only if n=0 and we say that the claim is true $\forall n \epsilon Z^+$ and thus we need to complete a base case $\forall n \epsilon Z^+$.

2) The claim is **false** - the proof is incorrect because ∃p1,p2,p3 s.t. they are not on the same plain. for example : (1,0) , (2,0) , (1,2) .

what went wrong ? In the induction step it is written that if $l_1$and $l_2$share the same points p2,...,pn then $l_1 = l_2$.

but when n=3, $l_1$and $l_2$ share only one point and when two lines share only one point it is not nesserally that they are on the same plain. (they can be crossed or vertical).

thus when n=3 this claim does not hold.

3) The claim is **true** - the base case holds for k=1 but also true when k≥1.

the induction assumption is true as well , and the algebraic transitions are correct in the induction step.

## Question C

**The value of x at the end of the run is 1** - we know that because we multiply x by 2 n times and divide the outcome by 2 n times, thus x stays the same, s.t. x=1.

**The value of r at the end of the run is 2n** - at first r=n, and then the number of times we run the first loop is the same as the second loop and equals to n.

Thus we add n elements to r on the second loop, and we obtain r = n+n s.t. r=2n.

**The value of s at the end of the run is $2+\sum_{i=1}^{n} \mathbf{i \cdot 2^i} = \mathbf{(n-1) \cdot 2^{n+1} + 4}$** - as we run the loops we know s is added to it self plus i*x.

We can represent the value of x at the end each iterartion as $2^i$ (since x is multipled by 2 even iterarion). and when

Now, when we multiply i by x n times we obtain $\sum_{i=1}^{n} \mathbf{i \cdot 2^i}$ , but we first defined s to be 2 thus we need to add 2 to the sum, and thus the expression is $\mathbf{2+\sum_{i=1}^{n} i \cdot 2^i}$.

In question #2 we proved this function is equal to $\mathbf{(n-1) \cdot 2^{n+1} + 2+2 = (n-1) \cdot 2^{n+1} + 4}$.

## Question D

1) **List Merge ($\mathbf{l_1}$,$\mathbf{l_2}$)** ////////////I assume that head[List] is the first node of the list.

NL = create_list
head1=head[$l_1$]
head2=head[$l_2$]
current←head[NL]
for i←1 to ($l_1$+$l_2$)
if ( data[head1] < data[head2] and data[next[head1]] ≠ null)
next[current] ← head1
current ←next[NL]
head1← next[head1]
end
if ( data[head1] > data[head2] and data[next[head2]] ≠ null)
next[current] ← head2

current ←next[NL]
head2← next[head2]
end
if (data[next[head1]]=null)
next[current] ← head2
current ←next[NL]
head2← next[head2]
end
if (data[next[head2]]=null)
next[current] ← head1
current ←next[NL]
head1← next[head1]
end
end of for loop
return NL

2)**List Merge2.0 ($l_1$,$l_2$)**
N=creat_node
NL2.0 = create_list
data[N]←data[head[$l_2$]]
head2=next[head[$l_2$]]
NL2.0←List Merge (N,$l_2$)
return List Merge (NL2.0,$l_1$)

## Question E

1)Array D[] - is representing the data of each node.
ArrayN[] - represent the index of the next node on the list.
Array P[] - represent the index off the prev node at the list.
head- the index of the first element of the list.
free- is the index of the next free element on the list.
n- the length of the lists.
for every i that is represented as each node of the list :
D[i] = the data on node[i].
N[i] = the index of the next element
P[i] = the index of the prev node.
when we have and index i that it is not on the list we have N[i] - the next free cell.
we also mention NIL=-1

2)Insert_Last(L,k)
if ( L.free =-1)
doublesize(L)
end
x = L.N [L.head]
while (L.N[x] ≠-1)
x←data[L.N[x]]
end while loop

first←L.N [L.head]
if (first = -1)
L.P.[first] ←-1
L.D. [first] ←k
L.free ←L.N.[free]
L.Nn [first] ←-1
end
else
y←x
L.D[y] ←k
L.P[y]←x
L.free ← L.N[y]
L.N[y] ←-1
L.N[x] ← L.free
end

doublesize(L)
NRR = create_new_array_size2n
for i←0 until i←n-1
NRR.P[i]←L.P.[i]
NRR.D[i]←L.D[i]
NRR.N[i]←L.N.[i]
end for loop
NRR.P[2n-1]←2n-2
NRR.N[2n-1]←-1
NRR.P[n+1]←-1
NRR.N[n+1]←n+2
for i←n until i←2n-2
NRR.P[i]←i-1
NRR.N[i]←i+1
end for loop
NRR.free←2n-1
return NRR

3) DeletePrevKey(L,k)
first←L.N[L.head]
if(first=-1)
return
end
i←first
while ( i≠-1 and L.D[i]≠k )
i ←L.N[i]
if( i =-1)
return the element does not exist
end if
end while loop
if ( L.N[i]≠-1 )

```
L.P[L.N[i]]←L.P[i]
end
if ( L.P[i]≠-1 )
L.N[L.P.[i]]←L.N.[i]
end
L.N.[i] ← L.free
L.P.[i] ←-1
L.P[L.free]←i
L.free ←i
end
```