Report for assignment 4

Shashwat Gupta (14IE10028)

February 3, 2016

1 Find order of people dying

I allow the user to input the number of people. I check if there are any inconsistencies in the input, in which case the code is immediately terminated. After this the input is passed to the **elimin_solve** function which finds and prints the order in which the people die.
Initially I pass the parameters start=1, n=input size and step=2.

The following is the Code of void elimin_solve

```
1  void elimin_solve(int start, int n, int step)
2  {
3      int printer,i;
4      if(n==1)
5      {
6          printf("\nThe survivor is : %d",start);
7          return;
8      }
9      printer=start+step/2;
10     for(i=1;i<=n/2;i++,printer+=step)
11         printf("%d\t",printer);
12     if(n%2==0)
13         elimin_solve(start,n/2,step*2);
14     else
15     {
16         printf("%d\t",start);
17         elimin_solve(start+step,n/2,step*2);
18     }
19 }
```

Listing 1: elimin_solve

This code is based on the following algorithm:

- It receives three parameters namely: start, n and step. **start** is the position from which the shooting begins. **n** is the number of people remaining. **step** is the amount by which the position advances after each shot. Here we shoot the adjacent alive person, thus initially 2 is shot followed by 4, 6, 8, etc. Thus the step initially is 2.
- There is a base case check for n=1 which indicates that there is only one person remaining, in which case that number is printed as the survivor and the recursion finally terminates.

- If the base case is not satisfied, then a variable called printer is used to print all the positions that will be shot in that round.
- The printer variable is initialised as start+step/2. This is because if I am starting at position 1 and I am moving at a step of 2, then the first person to die is position 2, followed by 4, 6, 8 and so on. Thus, whatever my step is, the initial position to be shot will always be start + step / 2 (initially it is 1 + 2/2 = 2, which is correct) and then this will increment by step (2+2=4, 4+2=6, etc). This incrementation only occurs n/2 times as after that we would exceed the start position.
- Then according to n being odd or even, the next recursive call is made and the start, n and step are set. For even, start stays same but n becomes half and step doubles. For odd, the start position dies thus we start with start+step and n becomes half and step doubles.

2 Recursively find Last man standing

We solve the problem when every 2nd person will be killed. We express the solution recursively.
Let f(n) denote the position of the survivor when there are initially n people.
The first time around the circle, all of the even-numbered people die.
The second time around the circle, the new 2nd person dies, then the new 4th person, etc.
If the initial number of people was even, then the person in position x during the second time around the circle was originally in position 2x - 1 (for every choice of x).
Let n=2j. The person at f(j) who will now survive was originally in position 2f(j) - 1.
This gives us the recurrence for even n:

$$f(2j) = 2f(j) - 1$$

Ex. Say for 4 people:
in round 1: 2, 4 died, in round 2: 3 died, survivor: 1.
now looking at the recursion, for n=1, f(1)=1, for n=2, f(2)=1 and f(2)=2f(1)-1 which holds.

If the initial number of people was odd, then we think of person 1 as dying at the end of the first time around the circle. Again, during the second time around the circle, the new 2nd person dies, then the new 4th person, etc. In this case, the person in position x was originally in position 2x+1.
This gives us the recurrence for odd n:

$$f(2j + 1) = 2f(j) + 1$$

Ex. Say for 5 people:
in round 1: 2, 4, 6 died, in round 2: 3, 1 died, survivor: 5

now looking at the recursion, for n=3, f(3)=3, for n=1, f(1)=1 and f(2*1+1)=2f(1)+1 which holds.

The following is the Code of func_solve

```
int func_solve(int n)
{
    int i;
    if(n==1)
        return 1;
    else if(n%2==0)
        return 2*func_solve(n/2)-1;
    else
        return 2*func_solve(n/2)+1;
}
```

Listing 2: func_solve

3 Analytically finding the Output

Check both analytically whether the survivor position can be obtained as:

$$V(2m+l) = 2l+1$$

$$0 \leq m, 0 \leq l < 2m$$

When we tabulate the values of n and f(n) we see a pattern:

n	1	2	3	4	5	6	7	8	9	10
f(n)	1	1	3	1	3	5	7	1	3	5

This suggests that f(n) is an increasing odd sequence that restarts with f(n)=1 whenever the index n is a power of 2. Therefore, if we choose m and l so that

$$n = 2^m + l, 0 \leq l < 2^m, f(n) = 2l+1$$

It is clear that values in the table satisfy this equation. Or we can think that after l people are dead there are only 2^{m} people and we go to the $2\text{l}+1^{\text{th}}$ person. He must be the survivor. So f(n)=2l+1. Below, we give a proof by induction.

Proof: We use strong induction on n. The base case n=1 is true. We consider separately the cases when n is even and when n is odd.

If n is even, then choose l1 and m1 such that

$$n/2 = 2^{m1} + l1, 0 \leq l1 < 2^{m1}, l1 = l/2$$

$$f(n) = 2f(n/2) - 1 = 2(2l1+1) - 1 = 2l+1$$

where the second equality follows from the induction hypothesis.

If n is odd, then choose l1 and m1 such that

$$(n-1)/2 = 2^{m1} + l1, 0 \leq l1 < 2^{m1}, l1 = (l-1)/2$$

$$f(n) = 2f((n-1)/2) + 1 = 2(2l1+1) + 1 = 2l+1$$

where the second equality follows from the induction hypothesis. This completes the proof.

We can solve for l to get an explicit expression for f(n):

$$f(n) = 2(n - 2^{\lfloor \log_2(n) \rfloor}) + 1$$

The following is the Code of check_print

```
int check_print(int n)
{
    int m=1;
    while(n>=m)
        m=m*2;
    m/=2;
    return(2*(n-m)+1);
}
```

Listing 3: check_print

The code takes as input n and then returns the position of the survivor. The result is printed and the user can compare this result with the programmatically obtained result.

4 Main

The following code is the main function that druns when the program is executed. The user is asked for the input. The inconsistency check is done and the relevant functions are called to get the required output.

The following is the Code of int main

```
int main()
{
    int n;
    printf("Enter the number of people (n) : ");
    scanf("%d",&n);
    if(n<=0)
    {
        printf("Incorrect inputs, program terminated\n");
        exit(0);
    }
    elimin_solve(1,n,2);
    printf("\nFinal Survivor from recursive function: %d\n",
    func_solve(n));
    printf("Final Survivor using \"V(2m+1)=2l+1\" : %d\n",
    check_print(n));
}
```

Listing 4: main

5 Time Complexity Calculation

Time complexity has the recursive relation:

$$T(n) = T(n/2) + \theta(n)$$

Solving this equation by recursive method yields T(n) $\in O(n)$.