Homework 11 by Timofei Podlorytov

11.1

a)

<3, 10, 2 ,4> _ _ _ _ _

First lets calculate the values of the hash function for all elements as well as the formula for h(k,i)

element(k)	$h_1(k)$= k mod 5	$h_2(k)$= 7k mod 8	h(k,i)=($h_1(k)$+i*$h_2(k)$)
3	3	5	3+5i
10	0	6	0+6i
2	2	6	2+6i
4	4	4	4+4i

Now we apply the function

k	i	h(k,i)	Hash table	collisions
3	0	3	Nil Nil Nil Nil Nil	none
10	0	0	Nil Nil Nil 3 Nil	none
2	0	2	10 Nil Nil 3 Nil	none
4	0	4	10 Nil 2 3 Nil	none
End:			10 Nil 2 3 4	

b)

the solution to the problem is in the provided cpp file

I selected the function for hashing to be double hashing as it reduces the chances of clustering of the values that might have the same hash value. And for both hash function I chose large enough prime numbers since it means that there will be more outputs.

11.2

Sorted according to length:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1																
											2					
3																
				4												
								5								
												6				

Lets' s say we sort these tasks according to the length of the task.
Then we choose the first option for which the start is after the end of the previous task we already chose.(locally optimal solution). And thus go through the list once.

If we follow this logic we'll get {1,2}. However, the best solution would be {3,4,5,6} or {1,4,5,6}. Meaning that the algorithm is not working.

Proper sorting according to the end time of the task:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1																
3																
				4												
								5								
											2					
												6				

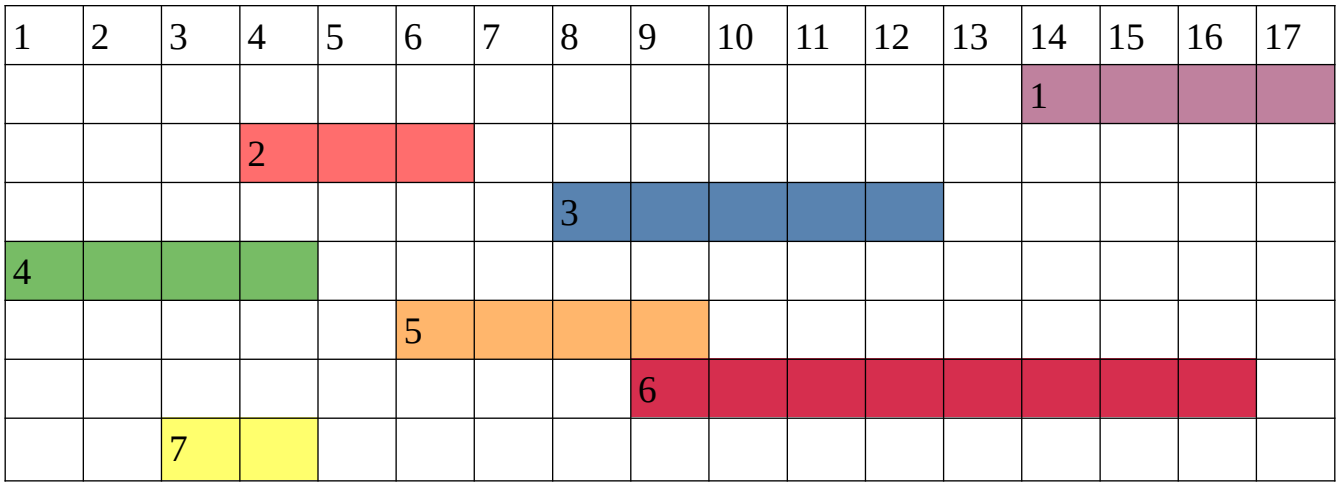
Proper algorithm would result in {1,4,5,6} of length four which is optimal.

b)
n-number of activities
A=an activity that has a start and an end

```
//this is our function
ActivitySelection(n, A):
        unSelected=A//we copy all the activities
        selected={}//a place for the result
        previousStart=+inf;//previous start
        while unSelected.size>0:
                latest=-1;//we search for latest activity we can use
                latestActivity=Nil;
                for activity in  A: //we search for the fitting activity with latest start
                        if activity.end>previousStart:
                                unSelected.remove(activity);
                        if (latest< activity.start):
                                latest=activity.start;
```

```
                    latestActivity=activity;
            selected.add(latestActivity);
            previousStart=latestActivity.start;
            unSelected.remove(latestActivity);
    return Selected;
```

We basically copy all the elements to a new activity array. While there is still something there we go through all activities left not selected and if it can't be used(finishes before the next starts) we delete it and otherwise get the latest start and add it to the list of selected activities, while removing it from the not selected list.



Let's say we were given these tasks. The result of the algorithm would be:

unSelected={1,2,3,4,5,6,7} Selected={}
unSelected={2,3,4,5,6,7} Selected={1}
unSelected={2,4,5,7} Selected={1,3}
unSelected={4,7} Selected={1,3,2}
unSelected={} Selected={1,3,2}

The answer is {1,2,3} which is optimal as its length is 3