Hashing Group Assignment

Group No: 27

Name	Index	Registration Number	Email Address
H.Y.H.P.Gunathilaka	17020281	2017/IS/028	yasika14@gmail.com
M.P.Pasyala	17020638	2017/IS/063	manujapasyala95@gmai.com
Weerasinghe W .A.S.J	17020964	2017/IS/096	sachiejayen@gmail.com

Question 01

Test case	Test case description	Expected result	Actual result	Pass / Fail
1	The test data is inserted to the code.	Display the maximum subarray sum is 23 with the highlighted elements.	Displays 23	Pass

Net beans IDE used in implementing this code. Do basically java is the programming language that used. When implementing the code I inserted the test data to the code. The array is divided into two sides from the middle and get the maximum from both the sub arrays.

IMAGE ONE

```
static int Get_max_Array(int arr[], int left,int high)
{
 if (left == high)
    return arr[left];

 int mid = (left + high)/2;

 return Math.max(Math.max(Get_max_Array(arr, left, mid),Get_max_Array(arr, mid+1, high)),maxSum(arr, left, mid, high));
}
```

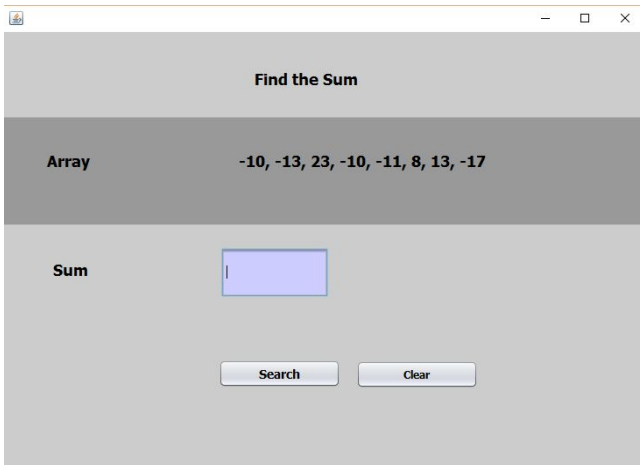
```
static int maxSum(int arr[], int left,int mid,int high){
int total = 0;
int left_sum = Integer.MIN_VALUE;
for (int i = mid; i >= left; i--)
{
    total = total + arr[i];
    if (total > left_sum)
    left_sum = total;
}
total = 0;
int r_sum = Integer.MIN_VALUE;
for (int i = mid + 1; i <= high; i++)
{
    total = total + arr[i];
    if (total > r_sum)
    r_sum = total;
}
return left_sum + r_sum;
```

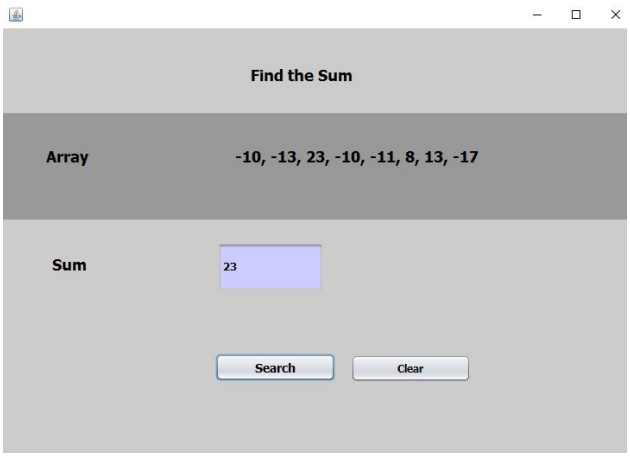
- Then the sum is displayed after getting the maximum of each sub array.

IMAGE TWO

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

        int arr[] = {-10, -13, 23, -10, -11, 8, 13, -17};
        int n = arr.length;
        int maximum_sum = Get_max_Array(arr,0,n-1);
        String s = Integer.toString(maximum_sum);
        Sum.setText(s);

}

/**
```

Screen Shots





Question 02

Test case	Test case description	Test steps	Expected result	Actual result
1	There is an array of independent jobs where every job is associated with a deadline and profit. The profit could be claimed only if the job is finished before the deadline.	Run the program then input job count and then job data.	Get the maximize total profit if only one job can be scheduled at a time.	Answer is same as the expected result.

```
void jobSchedule(Job arr[], int n){
    //Sort all jobs according to decreasing order of profit
    sort(arr, arr+n, comparison);

    int jobSeq[n]; //Sequence of jobs
    bool slot[n]; //track free time slots

    // Initialize all slots to be free
    for (int i=0; i<n; i++)
        slot[i] = false;


    for (int i=0; i<n; i++){
        int nxt=min(n, arr[i].dedL)-1;
        for (int j=nxt; j>=0; j--){
            if (slot[j]==false){
                jobSeq[j] = i;
                slot[j] = true;
                break;
            }
        }
    }
    int totprof=0;

    for (int i=0; i<n; i++){
        if (slot[i]){
            cout << arr[jobSeq[i]].id << " ";
            cout << arr[jobSeq[i]].profit<<endl;
            totprof=totprof+arr[jobSeq[i]].profit;
        }
    }
    cout<<"total profit : "<< totprof;
}
```

Here we use the Greedy Algorithm technique to solve this problem. In the code, first we sort all jobs in decreasing order of profit.Then we initialize the result sequence as first job in sorted jobs. And that is continuously followed for the remaining n-1 jobs. Here if the current job can fit in the current result sequence without missing the deadline, add current job to the result. Else ignore the current job. This is done using the function jobSchedule. Finally the jobs will be printed according to their maximum profit values and their subtotal is printed at the end.

Time complexity of this solution is $O(n^2)$.

Result Data