# Experiment 10

(Greedy and Bound approach)

## TASK - 1

**NAME: SATYAM**

**UID : 20BCS9393**

**CLASS: 607A**

**SUBJECT: CC LAB**

1. Aim:

Marc's Cakewalk

2. Code :

```
#include <bits/stdc++.h>


using namespace std;


string ltrim(const string &);
string rtrim(const string &);
vector<string> split(const string &);


long  marcsCakewalk(vector<int>  calorie)
{         sort(calorie.begin(),  calorie.end());
int cakes = calorie.size();
   int x = 0;     long miles = 0;
for(int i = cakes-1; i >= 0; --i) {
       miles += calorie[i] * pow(2, x);
x++;
   }
   return miles;
}

int main()
{
   ofstream fout(getenv("OUTPUT_PATH"));
```

```
    string  n_temp;          getline(cin,
n_temp);                     int     n     =
stoi(ltrim(rtrim(n_temp)));

    string calorie_temp_temp;
    getline(cin, calorie_temp_temp);


    vector<string> calorie_temp = split(rtrim(calorie_temp_temp));
vector<int> calorie(n);     for (int i = 0; i < n; i++) {
        int calorie_item = stoi(calorie_temp[i]);
calorie[i] = calorie_item;
    }
    long result = marcsCakewalk(calorie);
    fout << result << "\n";
fout.close();
    return 0;
}
string ltrim(const string &str) {
    string s(str);

    s.erase(
        s.begin(),         find_if(s.begin(), s.end(),
not1(ptr_fun<int, int>(isspace)))
    );
    return s;
}
string rtrim(const string &str) {     string s(str);     s.erase(
find_if(s.rbegin(), s.rend(), not1(ptr_fun<int, int>(isspace))).base(),
s.end()
    );
    return s;
}
vector<string> split(const string &str) {
vector<string> tokens;     string::size_type start = 0;
string::size_type end = 0;     while ((end = str.find("
", start)) != string::npos) {
tokens.push_back(str.substr(start, end - start));
        start = end + 1;
    }
    tokens.push_back(str.substr(start));
return tokens;
}
```

3. Output :


Compilation Successful :)
Click the Submit Code button to run your code against all the test cases.
Input (stdin)
1 3
2 Ab1
Your Output (stdout)
1 3