Experiment 3.2

Student Name: UTKARSH JOSHI **UID: 21BCS9158**
Branch: CSE **Section/Group: 802-A**
Semester: 5th **Date of Performance:14/11/2023**
Subject Name: Advanced Programming Lab-1 **Subject Code: 21CSP-314**

1. Aim:

Implement the problem based on backtracking.

2. Objective:

I. **WAP to check binary palindrome.**
II. **WAP to solve a crossword puzzle.**

3. Code:

i. **WAP to construct the array.**

```
#include <bits/stdc++.h>
using namespace std;
typedef long long int ll;

#define mp make_pair
#define pb push_back
#define pob pop_back()
#define mod 1000000007
#define max INT_MAX
#define min INT_MIN
#define fi first
#define se second
#define fast_cin() ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL)

set<int> v;
void binarypalindrome(int s, int e, int x)
{
   if(s>e)

   {v.insert(x);
```

```
    return;}
    binarypalindrome(s+1,e-1,x);
    if(s==e)
    binarypalindrome(s+1,e-1, x+ pow(2,s));
    else
    binarypalindrome(s+1,e-1, x+ pow(2,s)+ pow(2,e));
    return;
}
int main()
{
    fast_cin();
    int n,t;
    v.insert(0);v.insert(1);v.insert(3);
    for(int i=3; i<32; i++)
    {
        int c= pow(2,i-1)+1;
        binarypalindrome(1,i-2, c);
    }
    cin>>t;
    while (t--)
    {
        cin>>n;
        auto ptr=v.lower_bound(n);
        auto ptr2= ptr--;
        if(abs(n-*ptr)<abs(n-*ptr2))
        cout<<abs(n-*ptr)<<endl;
        else
        cout<<abs(n-*ptr2)<<endl;
}
return 0;
}
```

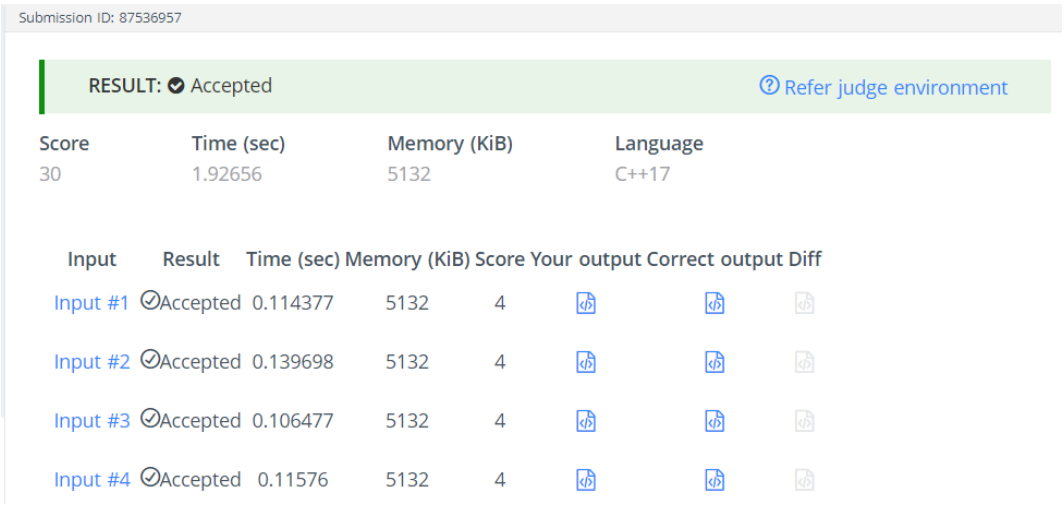
ii. WAP to minima the operation on array.

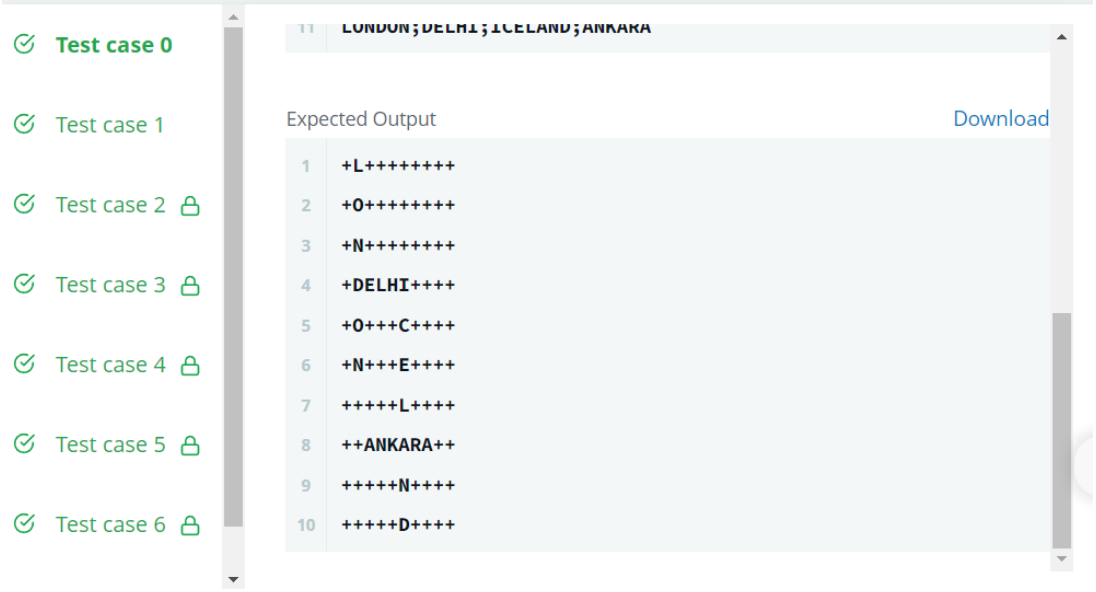
```
#include <bits/stdc++.h>
using namespace std;
```

```
#define speed ios_base::sync_with_stdio(0); cin.tie(0); cout
.tie(0)
#define pb push_back
#define sz size()
#define cin(v) for (auto& i : v) cin >> i
vector<string> g(10), words;
vector<int> vis(10);
bool done = false;
void solve(int i) {
    if (done) return;
    if (i == 10) {
        if (accumulate(vis.begin(), vis.end(), 0) == words.s
z && !done) {
            done = true;
            for (auto a : g) cout << a << '\n';
        }
        return;
    }
    for (int j = 0; j < 10; j++) {
        if (g[i][j] == '+') continue;

        for (int k = 0; k < words.sz; k++) {
            if (vis[k]) continue;
            vis[k] = 1;
            vector<int> added;
            int c = 0;
            for (int m = 0; m < words[k].sz; m++) {
                int ind = m + j;
                if (ind >= 10 || g[i][ind] == '+') break;

                if (g[i][ind] == '-') {
                    g[i][ind] = words[k][m];
                    added.pb(ind);
                } else if (g[i][ind] != words[k][m]) break;
                else c++;
            }
            if (added.sz + c == words[k].sz) solve(i);

            for (auto ind : added) g[i][ind] = '-';
```

```
                added.clear();
                c = 0;

                for (int m = 0; m < words[k].sz; m++) {
                    int ind = m + i;
                    if (ind >= 10 || g[ind][j] == '+') break;

                    if (g[ind][j] == '-') {
                        g[ind][j] = words[k][m];
                        added.pb(ind);
                    } else if (g[ind][j] != words[k][m]) break;
                    else c++;
                }
                if (added.sz + c == words[k].sz) solve(i);

                for (auto ind : added) g[ind][j] = '-';

                vis[k] = 0;
            }
        }
        solve(i + 1);
    }
    void runCase() {
        cin(g);
        string s;
        cin >> s;
        stringstream ss(s);
        string x;

        while (getline(ss, x, ';')) words.pb(x);

        solve(0);
    }
    int main() {
        speed;
        runCase();
    }
```

4. Output:





5. Learning and Outcomes:

- o Concept of backtracking.
- o Improve problem solving skills.
- o To understand how to recursive thinking.