DAA

Assignment no. 4

Name: Aditya Rajesh Jadhav.

Reg.no.: 2020BIT044

1] Traveling salesman problem:

Code ->

```
#include <bits/stdc++.h>
using namespace std;
#define vr 4
int TSP(int grph[][vr], int p){ // implement traveling Salesman Problem.
  vector<int> ver;
  for (int i = 0; i < vr; i++)
    if (i != p)
      ver.push_back(i);
      int m_p = INT_MAX; // store minimum weight of a graph
  do {
    int cur_pth = 0;
    int k = p;
    for (int i = 0; i < ver.size(); i++) {
      cur_pth += grph[k][ver[i]];
      k = ver[i];
    }
    cur_pth += grph[k][p];
    m_p = min(m_p, cur_pth); // to update the value of minimum weight
```

```
  }
  while (next_permutation(ver.begin(), ver.end()));
  return m_p;
}
int main() {
  int grph[][vr] = { { 0, 5, 10, 15 }, //values of a graph in a form of matrix
    { 5, 0, 20, 30 },
    { 10, 20, 0, 35 },
    { 15, 30, 35, 0 }
  };
  int p = 0;
  cout<< "\n The result is: "<< TSP(grph, p) << endl;
  return 0;
}
```

Output ->

```
PS D:\c c++> cd "d:\c c++\DAA_ass_4\"
ling_salesman_problem }

 The result is: 75
```

2]String matching :

Code ->

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

void search(char* pat, char* txt)
{
	int M = strlen(pat);
	int N = strlen(txt);

	/* A loop to slide pat[] one by one */
	for (int i = 0; i <= N - M; i++) {
		int j;

		/* For current index i, check for pattern match */
		for (j = 0; j < M; j++)
			if (txt[i + j] != pat[j])
				break;

		if (j
			== M) // if pat[0...M-1] = txt[i, i+1, ...i+M-1]
			cout << "Pattern found at index " << i << endl;
	}
```

```
}

// Driver's Code
int main()
{
        char txt[] = "AABAACAADAABAAABAA";
        char pat[] = "AABA";

        // Function call
        search(pat, txt);
        return 0;
}
```

Output ->

```
PS D:\c c++\DAA_ass_4> cd "d:\c c++\DAA_ass_4\"

Pattern found at index 0
Pattern found at index 9
Pattern found at index 13
```

3] Exhaustive search :

Code ->

```
#include<iostream>
using namespace std;

int exhaustive_search(int head,int foot,int *chicken,int * rabbit)
{
   int re,i,j;re=0;
   for(i=0;i<=head;i++)    //cycle
   {
      j=head-i;
      if(i*2+j*4==foot)   //judge
     {
         re=1;   //To find the answer
         *chicken=i;
         *rabbit=j;
      }
   }
   return re;
}
int main()
{
int chicken,rabbit,head,foot;
int re;
cout<<" The exhaustive method is used to solve the problem: "<<endl;
```

```
cout<<" Please enter the number of heads: ";
cin>>head;
cout<<" Please enter the number of feet: ";
cin>>foot;
re=exhaustive_search(head,foot,&chicken,&rabbit);
if(re==1)
{
  cout<<" The chicken has "<<chicken<<" Is the only, rabbit "<<rabbit<<" Only.
"<<endl;
}
else
{
  cout<<" Unsolvable! "<<endl;
}
return 0;
}
```

Output ->

```
PS D:\c c++\DAA_ass_4> cd "d:\c c++\DAA_ass_4\" ; if ($?)
arch }
 The exhaustive method is used to solve the problem:
 Please enter the number of heads: 12
 Please enter the number of feet: 8
 Unsolvable!
```