Assignment - IV

Name := G.Harshitha

Registration No := 192324250

Department := AI & DS

Course name := DATA STRUCTURE

Course code := CSA0389

Starting Date := 03-08-2024

Submission Date := 05-08-2024

① Perform the following operations using stack Assume Size of the stack is 5 and having a value of 22, 55, 53, 33, 66, 88 in the stack form a position to size-1 Now perform the following Operations :

i) Invert the elements in the stack 2) pop() 3) pop() 3pop(), 3) pop(), 4) pop(). Draw the diagram of stack and illustrate the above Operations and identify where the top is ?.

Ⓐ Implementation of the stack :=

```
#include <stdio.h>
#define MAX-size 5
type def struct{
    int data [MAX-size];
    int top;
    }
    stack;
void in stack (stack * s){
    s -> top = -1;
    }
int is empty (stack * s){
return s -> top = -1;
}
int is full (stack *s){
return s -> top = MAX-size -1;
}
void push (stack *s, int value)
if (is full (s)){
Printf("stack is full. cannot push %. d %. n", value);
```

```
return;
 }
 s->data[++s->top] = value;
}
int top(stack *s){
  if(isempty(s)){
Printf("stack is empty. cannot pop.\n");
  return -1
  }
  return s->data[s->top--];
}
void invert[stack *s]{
  int temp[max-size];
   int i,j;
 for(i=0;j=s->top;i<=j;i++,j--){
  temp[i] = s->data[j];
  temp[j] = s->data[i];
  }
 for(i=0;i<=s->top;i++)
    s->data[i] = temp[i];
  } }
  int main(){
 Stack s;
 Push(&s,22);
 Push(&s,55);
 Push(&s,33);
 Push(&s,66);
 Push(&s,88);
```

```
Printf("Initial stack :\n");
 Printstack (&s);
   invert (&s);
Printf("After inverting :\n");
 Printstack(&s);
 Printf("popped : %d \n", pop(&s));
Printf("popped : %d \n", pop(&s));
Printf("popped : %d \n", pop(&s));
    push(&s,90);
    push(&s,36);
    push(&s,11);
    Push(&s,88);
   Printf("After pushing :\n");
   Printf Stack(&s);
 Printf("popped : %d \n", pop(&s));
 Printf("popped : %d\n", pop(&s));
  Print Stack(&s);
    return 0;
    }
```

Output :=

```
Initial stack :=
 stack  : 22  55  33  66  88
After Inverting : 88  66  33  55  22
   Popped  : 22
   Popped  : 55
   Popped  : 33
```

After pushing :-

Stack : 88 60 90 36 11

Popped : 11

Popped : 36

-final stack :=

Stack : 88 66 90

2. Develop an algorithm to detect duplicate elements in an unsorted array using linear search. Determine the time complexity and discuss how you would optimize this process.

A) To detect duplicate elements in an unsorted array using linear search.

```
#include <stdio.h>
void detect duplicates (int arr[], int n){
for (int i=0; i<n; i++){
 if[arr[i] = i+1; j<n; j++){
   if [arr[i] =arr[j]){
Printf("Duplicate element found : %d \n", arr[i]);
  return ;
  }
 }
 }
Printf("No Duplicate element found.\n");
 }
int main(){
int arr[] = { 5,2,8,12,3,2,1};
int n = size of (arr) (size of [arr(0)];
Detect Duplicates (arr, n);
return 0;
 }
```

Time Complexity :- The time complexity of this algorithm is $O(n^2)$ where n is the no. of elements in array. This is because using two nested loop to compare each element.

Optimized Version :-

```
#include<stdio.h>
#include<stdlib.h>
type def of struct{
  int *data;
  int size;
 }Harshtable;
Hash table *create hashtable (int size){
Hash table *ht = (Hash table*) malloc(size of (hash table
ht ->data = (int*)malloc (size*size of (int));
    ht->size = size;
      return ht;}
 void insert (hashtable *ht, int value){
  int index = value %. ht ->size;
 while (ht ->data [index] ≠0){
 if(ht ->data[index] = value){
Printf("Duplicate Element found : %d\n");
 return ;}
 index = (index +1) % ht ->size;}
  ht -> data(index) = value ;}
   int main(){
  int arr( ) = { 5, 2, 8, 12, 3, 2, 1};
int n = size of (arr) [size of (arr[0]);
 delect duplicates (arr, n);
    return 0;
     }
```