Assignment-3

Name : CH. Naga Anjali

Reg no : 192372201

Dept : CSE-AI

Course name : Datastructure for stack overflow

Course code : CSA0389.

Perform the following operations using stack. Assume size of the stack is 5 and having a value of 22, 15, 33, 66, 88 in the stack form a position to size-1 Now perform the following operations:

(i) Quvert the elements in the stack 2) pop() 3) pop() 3pop(), 4) push(90), 5) push(36), 6) push(11), 7) push(88), 8) pop() 9) pop(). Draw the diagram of stack and illustrate the above operations and identify where the top is?

A. Implementation of the Stack:-

```
#include <stdio.h>
#define MAX_SIZE 5
typedef struct {
  int data [MAX_SIZE];
   int top;
 }
 stack;
void instack (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 pop (stack * s) {
if (is empty (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");
   Print stack (&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");
   Print 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 6688

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

Develop an algorithm to detect duplicate elements in an unsorted array using liuear 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 dulplicates (int arr[], int n) {
 for (int i=0; i<n; i++){
 for (int j = 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>
typedef struct {
int * data;
int size;
} Hash table;
Hash table * create Hash table (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 (hash table * 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]);
detect duplicates (arr, n);
return 0;
}
```