Assignment - 3

Name :- J. Surendra

Regd No :- 192324050

Subject :- Data structure

Sub code :- CSA0389

Date of Submission :- 05/08/24.

(1) Perform the following operations using stack. Assume size of the stack is 5 and having a value of 22, 55, 33, 66, 88 in the stack from a position to size-1 Now perform the following operations.

1. Insert the element in the stack 2) pop() 3) pop()

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?

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 isfull (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(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");
Print&tack(&s);
invert(&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");
    Printstack(&s);
    Printf("popped: %d\n", pop(&s));
    Printf("popped: %d\n", pop(&s));
  Printf 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 would optimize this Process.

To determine 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++){
for(int j=i+1;j<n;j++){
if(arr[i]=arr[j]){
Printf("Duplicate element found: %d\n", arr[i]);
return;
}
}
}
Printf("NO Duplrcates found:\n");
}
int main(){
int arr[]={5,2,8,12,3,2,1};
int n=sizeof(arr)/sizeof(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;
} Hashtable;
Hash table * create Hash table (int size){
Hash table * nt = (Hashtable* ) malloc (size of (hash tabli));
 ht -> data = (int*) malloc (size * size of (int));
 nt -> size = size;
 return nt; }
 void insert (hashtable * nt, int value){
 int index = value % nt + size;
 while (ht -> data [index] ≠ 0) {
 if (nt -> data [index] = value)
Printf ("Duplicate element found: %d \n");
return; }
index = (index + 1) % nt -> size; }
nt -> 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;
}.
```