04/09/2024

Ch. Nagi Reddy

Programming Test - 1

192324170

1] Array insertion, deletion

```
#include <stdio.h>
Void insertElement(int arr[], int n, int pos, int value) {
for (int i=n ; i>pos ; i--) {
    arr[i] = arr[i-1]
}
}
Void deleteElement (int arr[], int n, int pos){
    for (int i=pos ; i<n-1 ; i++) {
        arr[i] = arr[i+1]
    }
}
int main() {
int arr[100] = { 1,2,3,4,5 };
int n = 5
insertElement(arr, n, 2, 99);
n++;
Printf("Array of insertion:");
for (int i=0 ; i<n ; i++) {
  Printf("%d", (arr[i]));
}
  Printf("\n");
  deleteElement (arr, n, 3);
  n--;
  Printf("Array of deletion:");
  for (int i=0; i<n; i++) {
```

4/9/24

Output:-

Array after insertion:

1 2 99 3 4 5

Array after deletion:

1 2 99 4 5

```
    Printf(" %d", arr[i]);
  }
  return 0;
}
```

2. Stack using Array implementation

```
#include<stdio.h>
#define MAX 100
struct stack{
int arr[MAX];
int top;
};
void initialize(struct stack *s){
    (*s).top = -1;
}
int isFull(struct stack s){
    return s.top == MAX-1;
}
void push(struct stack *s, int value){
    if (isFull(*s)){
        printf("Stack Overflow\n");
        return;
    }
    (*s).arr[++((*s).top)] = value;
    Printf("%d Pushed to stack\n", value);
    return -1;
int pop(struct stack *s){
    if (isEmpty(*s)){
        Printf("stack Underflow\n");
    return -1;
```

```
    return (*s).arr[(*s).top--];
}
int peek(struct stack s){
    if(isEmpty(s)){
        Printf("stack is empty\n");
        return -1;
    }
    return s.arr[s.top];
}
void display(struct stack s){
    if(isEmpty(s)){
        Printf("stack is empty\n");
        return;
    }
    for(int i=s.top; i>=0; i--){
        Printf("%d", s.arr[i]);
    }
    Printf("\n");
}
int main(){
    struct stack s;
    initialize(&s);
    Push(&s,10);
    Push(&s,20);
    Push(&s,30);
    Printf("Top element is %d\n", peek(s));
    Printf("Elements in stack:");
    display(s);
    return 0;
}
```

Output:

```
10 Pushed to stack
20 Pushed to stack
30 Pushed to stack
30 Popped from stack
Top element is 20
Stack elements:
20
10
```

3.

```
#include<stdio.h>
#define MAX 5
struct Queue {
    int items[MAX];
    int items[MAX];
    int front, rear;
};
void initializeQueue(struct Queue *q) {
    q->front = -1;
    q->rear = -1;
}
int isfull(struct Queue *q) {
    return q->rear == MAX-1;
}
int isEmpty(struct Queue *q){
    return q->front == -1;
}
int dequeue(struct Queue *q) {
    int value;
    if (isEmpty(q)) {
        Printf("Queue is Empty\n");
        return -1;
    } else {
        value = q->items[q->front];
        q->front ++;
        if (q->front > q->rear) {
            q->front = q->rear = -1; }
        return value;
    }
}
int main() {
    struct Queue q;
    initialize Queue(&q);
    enqueue(&q, 10);
    enqueue(&q, 20);
    enqueue(&q, 30);
    display(&q);
    Printf("Removed: %d\n", dequeue(&q));
    return 0; }
```

output:
Inserted 10
Inserted 20
Inserted 30
Inserted 40
Inserted 50
Queue elements: 10 20 30 40 50
Removed: 10
Removed: 20
Queue elements: 30 40 50