제품 계열 방법론

특정 제품에 적용하고픈 **공통 기능 정의하여 개발**

~ 임베디드 소프트웨어

영역공학 : 영역분석 → 설계 → 핵심자산 구현
↓ ↓ 컴포넌트
제품요구사항 ↔ 아키텍처 ↔ 조립 생산
↓ ↓
응용공학 : 요구 분석 → 제품설계 → 제품구현

S/W 공학의 발전적 추세

소프트웨어 재사용

합성 중심 ((composition - Based)) : 블록을 만들어 끼워 맞춰 소프트웨어 완성시킴

생성 중심 (Generation - Based) : 명세를 구체화

소프트웨어의 재공학

새로운 요구에 맞게 기존 시스템 이용해 더 낫게 시스템 구축

분석 (analysis)

재구성 (restructuring)

역공학 (reverse engineering) : 기존을 분석하여 개발과정, 데이터처리과정을 설명&분석하여 다시..

이식 (migration)

CASE (Computer Aided Software Engineering)

요구되는 요구분석, 설계, 구현 등을 컴퓨터나 전용 소프트웨어 도구사용하여 자동화

공통모듈 사용함으로써 재사용 ↑ , 모듈관리 자동 수행

주요기능 : 생명주기 연결, 소프트웨어 개발 모형 지원, 모델 모순 검사, 오류 검증 자료흐름도 작성

원천기술 : 구조적기법, 프로토타이핑, 자동 프로그래밍, 정보저장소, 분산처리

비용산정기법 — 상향식

세부적 작업 단위별로 비용 산정후 집계하여 전체 비용 산정

① LOC (원시 코드라인 수, Source Line of Code) 기법

각 기능의 원시코드라인 수의 비관치, 낙관치, 기대치 측정후 예측치 구하고 이를 이용하여 비용산정

$$\text{예측치} = \frac{a+4m+b}{6}$$

(a : 낙관치, 가장 낙관적 측정 / m : 기대치, 평균 / b : 비관치, 가장 비관적 측정)

② 개발단계별 인월수 (Effort Per Task) 기법

LOC 기법 보완, 기능구현하는데 필요한 생명주기 단계별 산정

수학적 산정 기법

상향식 비용산정 기법 .. 경험, 실험 추정 모형 ~개발 비용산정

COCOMO 모형

LOC (원시 코드라인 수)에 의해 비용산정 기법

<유형>

- 조직형 (Organic mode)

  5만라인 이하 소프트웨어 개발하는 유형

  노력(MM) = $2.4 \times (KDSI)^{1.05}$

  개발기간(TDEV) = $2.5 \times (MM)^{0.38}$

- 반분리형 (Semi-Detached Mode)

  조직형 & 내장형 중간, 트랜잭션처리 시스템, 운영체제, 데베 관리 시스템 등

  30만 이하 소프트웨어 개발

  ~ 컴파일러, 인터프리터 같은 유틸리티 개발 용이

  노력(MM) = $3.0 \times (KDSI)^{1.12}$

  개발기간(TDEV) = $2.5 \times (MM)^{0.35}$

- 내장형 (Embedded Mode)

  초대형 규모..

  ~ 신호기 제어, 이사일 유도, 실시간 처리 시스템 개발

Puntam 모형

소프트웨어 생명주기 전과정동안 사용될 노력 분포 가정해주는 모형

~ Rayleigh-Norden 곡선

대형프로젝트의 노력분포 산정

$$MM = \frac{L^2}{C_k^3 \cdot Td^4}$$

기능점수(FP) 모형

기능 증대시키는 요인별로 가중치 부여, 합산하여 총 기능점수산출

FP = 총 기능점수 $\times$ [0.65 $\times$ (0.1 $\times$ 총영향도)]

* 자동화 측정도구

  SLIM : Rayleigh-Norden 곡선 & Puntam 예측모델 기초로 개발

  ESTIMACS : 다양한 프로젝트, 개인 요소수용 ~ FP 모형 기초..

IT 프로젝트 정보시스템 구축 관리

네트워크 관련 신기술

- 지능형 연결망

for 데이터 트래픽, 효과적 수용

국가전체망에 소프트웨어 정의 기술(SDE) 적용하는 방법

소프트웨어 정의 기술 (SDE ; Software-Defined Everything)

네트워크, 데이터 센터 등에서 소유한 자원을 가상화하여
사용자에게 제공하고 중앙에서 통합적으로 제어 가능한 기술

소프트웨어 정의 네트워킹

- 네트워크를 컴퓨터처럼 모델링하여 여러 사용자나 각 소프트웨어들로 네트워킹을 가상화하며 제어, 관리하는 네트워크
- 하드웨어 의존하는 것보다 효과적으로 네트워크 제어, 관리 가능
- 기존 네트워크에 영향 안주면서 전송경로 수집하며 발생하는 문제 처리 가능

소프트웨어 정의 데이터 센터

- 데이터 센터 자원 가상화하여 인력 개입없이 소프트웨어 조작만으로 관리, 제어되는 센터

소프트웨어 정의 스토리지

- 데이터 스토리지를 가상화하여 여러개를 하나로 관리하거나, 하나를 여러개로 나눠 사용하는 기술

IoT (Internet of Things, 사물 인터넷)

- 정보통신기술기반으로 실세계나 가상세계의 다양한 사물을 인터넷으로 연결하여 진보된 서비스 제공하기 위한 기술
- 유비쿼터스 공간 구현을 위한 컴퓨터 기기들이 환경, 사물에 상주 지능 통신을 할 수 있는 엠투엠 (Machine to Machine) 개념을 인터넷으로 확장하며 모든 정보와 상호작용하는 IoT 개념으로 진화

*-유비쿼터스

컴퓨터나 네트워크를 의식하지않고 자유롭게 네트워크 접속

- 개방형 아키텍처 ·· 보안기술 중요

클라우드 컴퓨팅 (Cloud Computing)

컴퓨팅자원을 중앙 컴퓨터에 두고 인터넷으로 작업을 수행할 수 있는 환경

중앙 컴퓨터 = 복수 데이터 센터를 가상화 기술로 통합한 데이터 기술

↔ 그리드 컴퓨팅 (Grid Computing) : 수많은 컴퓨터를 하나의 컴퓨터로 묶어 분산처리..

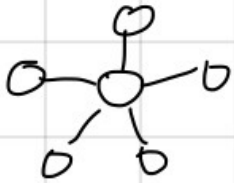
네트워크 구축

네트워크 설치 구조

. 정보전달을 위해 통신규약에 의해 연결한 통신설비 작업

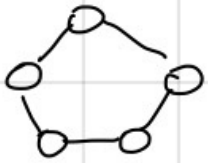
성형 (Star, 중앙집중형)

- 중앙에 중앙 컴퓨터, 중심으로 단말장치 연결되는 중앙집중식 네트워크 구성 형태 ⇒ 교환 노드수 적음.
- Point to Point 방식으로 회선연결
  - 중앙컴퓨터로 데이터 교환
  - 단말장치 추가/제거 쉬움



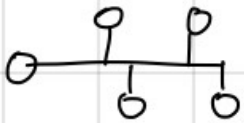
링형 (Ring, 루프형)

- 서로 이웃하는 것끼리 Point to Point 연결시킨 형태
  - 분산, 집중 제어 쉬움
  - 단말기 추가, 제거, 기밀보호 어려움
  - 전송지연, 중계기의 수 많아짐.
  - 데이터 단방향/양방향 전송



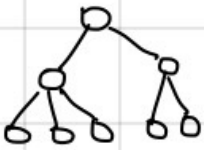
버스형 (Bus)

- 한개의 통신회선에 여러대의 단말장치 연결
- 물리적 구조 간단 & 단말기 추가&제거 용이
- 기밀보장 어렵고 통신회선 길이에 제한..



계층형 (Tree, 분산형)

- 중앙컴에 일정 지역까지 단말장치를 하나의 통신회선으로 연결
- 이웃하는 단말장치는 일정 지역내 설치된 중간 단말장치로부터 다시 연결
- 분산처리



망형 (Mesh)

- 모든 지점의 컴퓨터와 단말장치 서로 연결, 노드의 연결성 수
- 공중 데이터 통신망.. 통신 회선의 총 길이 가장 김
- 회선장애시, 다른 경로를 통해 데이터 전송가능
- 노드수: n , 회선망: $\frac{n(n-1)}{2}$, 포트 : 노드당 $n-1$개

네트워크 분류

근거리 통신망
(LAN ; Local Area Network)

- 가까운 거리 (학교, 회사..)
- 자원 공유
- 거리 짧아 전송속도 빠르고 에러 발생률 ↓
- 버스형, 성형

광대역 통신망
(WAN ; Wide Area Network)

- 먼 거리 (국가, 대륙)
- 통신 속도 ↓, 에러 발생률 ↑
- 근거리 통신망을 근거리통신망으로 연결

* VLAN (Virtual Local Area Network)
  LAN의 물리적 배선 상관없이 논리적으로 운용하는 기술

* CSMA / CA
  무선랜에서 데이터 전송시 아예 비어있음을 확인하고
  충돌 피하기 위해 일정시간 기다린후 전송

경로 제어

송·수신측간 전송경로 중에 최적 패킷 교환 경로 결정
어느한 경로에 데이터 양 집중 피함, 최저 비용, 최단시간 송신
경로 제어표 (Routing Table) 참조, 라우터에 의해 수행..

→ 경로 제어 프로토콜 (Routing Protocol)
효율적인 경로 제어를 위해 네트워크 정보 생성, 교환 제어하는 프로토콜

IGP (Interior Gateway Protocol)
- 하나의 자율시스템 (AS) 내 라우팅에 사용
- RIP (Routing information protocol)
  - 거리 벡터 라우팅 프로토콜
    ~최단 경로 탐색, Bellman - Ford 알고리즘
- OSPF (Open Shortest Path First Protocol)
  라우팅 정보에 노드간 거리, 링크 정보 실시간 반영
  ~다익스트라 알고리즘, 변화된 정보만 모든 라우터에 알림

EGP (Exterior Gateway Protocol)
자율시스템 (AS) 간 라우팅, 게이트웨이 간 라우팅에 사용

BGP (Border Gateway Protocol)
→ 보완.. 전체 경로 제어표 교환, 변화된 정보만 교환

트래픽 제어

전송되는 패킷 흐름/양 조절

흐름제어 (Flow Control)

송·수신측에 전송되는 패킷 양, 속도 규제

overflow 방지

- 정지-대기 (Stop-and-wait) : 수신측에 확인신호 받은후 다음 패킷 전송, 한번에 하나
- 슬라이딩-윈도우 (Sliding Window) : 송신 데이터 양 조절, 연속 전송

폭주제어 (Congestion Control)

네트워크내 패킷수 조절하여 네트워크 오버플로우 방지

- 느린 시작 (Slow start) : 2배수 증가
  임계값 도달시 혼잡 회피단계
- 혼잡회피 (Congestion avoidance) : 지수적 증가
  혼잡 → 선형적 증가

* dead lock : 공간 차지할 때

기억공간 할당기다림

[Java] Stack(스택) 사용법 및 예제

언어 Java 11 언어 설정

소스 코드 공개
- 공개
- 비공개
- 맞았을 때만 공개

소스 코드

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

import java.util.StringTokenizer;
import java.util.Stack;

class Main{
    public static void main(String [] args)throws IOException{
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

        int T = Integer.parseInt(br.readLine());

        String menu = "";
        StringTokenizer st;
        Stack <Integer> s = new Stack<Integer>();

        for(int i = 0; i<T; i++){
            st = new StringTokenizer(br.readLine());
            menu = st.nextToken();

            switch(menu){
                case "push":
                    s.push(Integer.parseInt(st.nextToken()));
                    break;
                case "pop":
                    if(s.size()==0)
                        System.out.println(-1);
                    else
                        System.out.println(s.pop());
                    break;
                case "size":
                    System.out.println(s.size());
                    break;
                case "empty":
                    if(s.size()!=0)
                        System.out.println(0);
                    else
                        System.out.println(1);
                    break;
                case "top":
                    if(s.size()==0)
                        System.out.println(-1);
                    else
                        System.out.println(s.peek());
                    break;
            }
        }
    }
}
```

제출