COP
Concept Of Programming.

14-3-22.

for C: →

install – devCPP
↳ Editor
→ compiler
↳ JDK/JRE

– VSCode
↳ Editor
↳ Compiler

for Java: →

install – Notepad / Eclipse
↳ Editor
↳ MinGW.

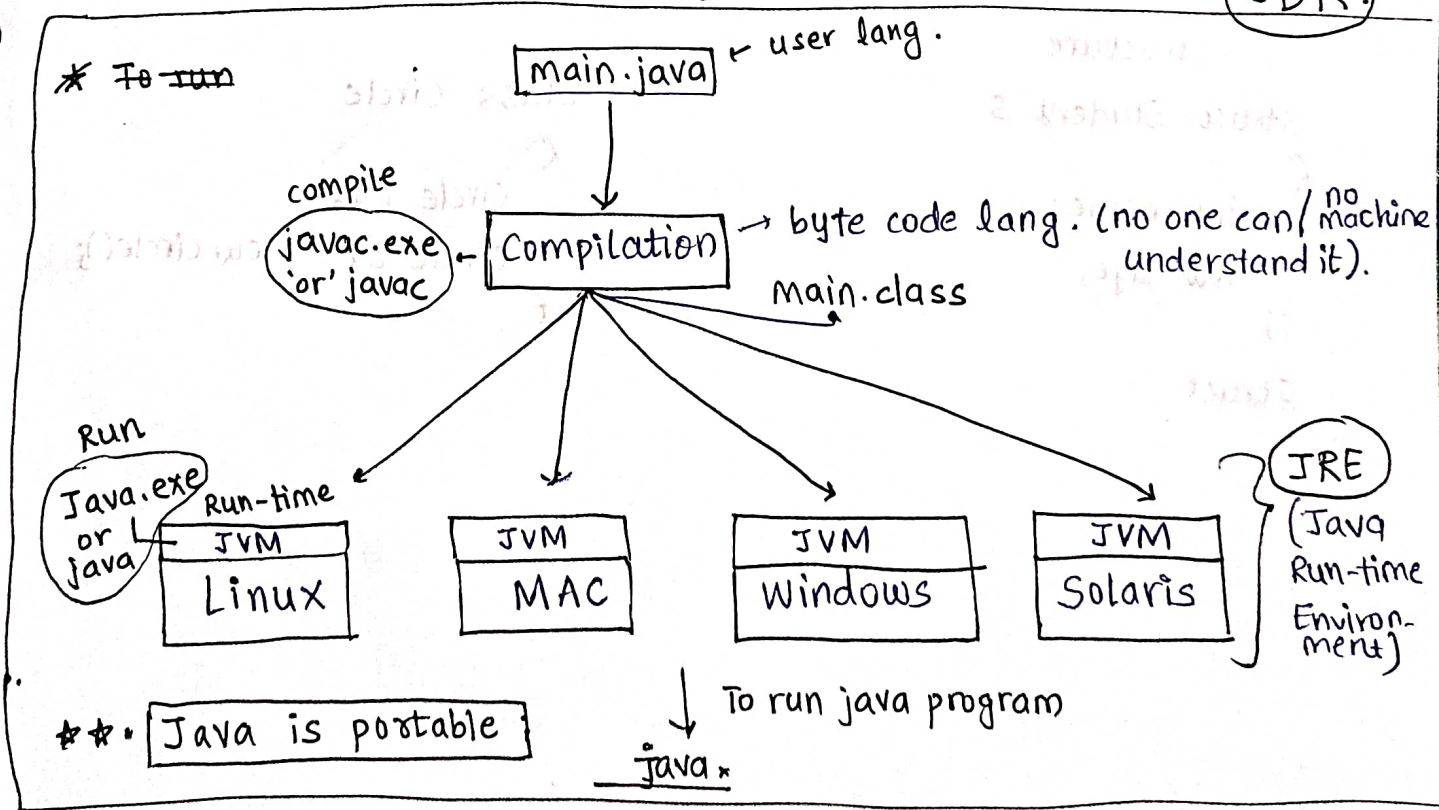
** To make a software, the tools we use is knowns as
Slw Development kit → SDK

** To make a Java app^n, the tools we use is known as.
Java Development kit → JDK

porting → C is machine ~~in~~dependent (i.e. h/w). → any h/w, windows or linux or mac

Java is machine independent (i.e. h/w independent)
↳ it can run on any machine.

JDK.



** Java compilation process →

15-3-22

* **JVM** - depending upon h/w.

i/p (byte code) - JVM - o/p (OP code)
operational code.

* **Class** - classes are bydefault typedefine structure.

Hungarian Notation - (for 2 words).

in C

int center_x; ~~class~~ (2 letters separated by _).

in Java (first letter small, 2nd capital)

int centerX

Entity → For whom you ~~return~~ have written class. (Noun) e.g. Girl.

Instance → ~~Attributes~~

e.g. attributes → name, Age, height, width, lipstick, Hairstyle, BP } **attribute**

the class for which we ~~have~~ use needed attributes is known as entity.

• we cannot write massless entity as class.

C

structure

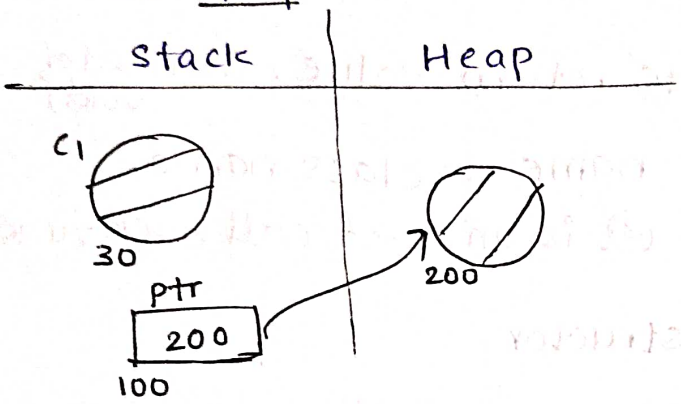
```
struct Student s
{
    int name;
    int age;
};
struct
```

Java

```
class Circle
{
    Circle c1;
    Circle c2 = new Circle();
}
```

C & C++

① • objects/variables are always created on stack.
• & also ~~cre~~ can be created on Heap.



② malloc
free();

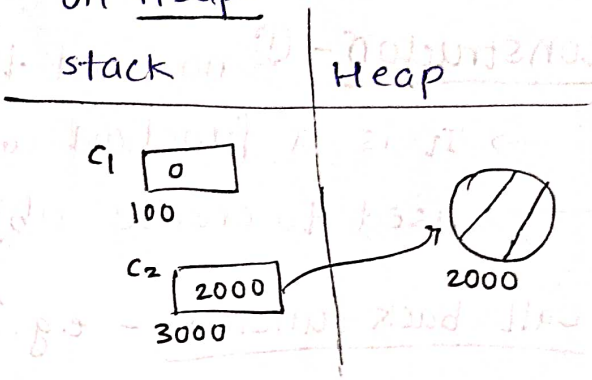
③ we have functions in C.

④ Pointer

⑤ Object

Java

① • Never allows object to create on stack.
• Object is always created on Heap.



② new object is created.
circle c2 = new circle();
Garbage collector. (GC)

③ we have methods in Java.

④ Reference.

⑤ Instance

Method Overloading — / Function Overloading -

→ same method but different parameters (arguments).

constructor - ① no need to write return value. (no need of void).

② → It is a function whose name = class name.

→ used to create object. ③. is an ex. of call back func^.

Call back function - e.g. ↑ constructor.

Circle-point, tostring, User-Developer, Array (string). 16/03/22

private-bublic, User-input

IQ. Default constructor → given by compiler with no parameters.
It is always not true.

- If any parameterized constructor is available, then compiler doesn't give default constructor.
- If their is not any constructor defined, compiler gives default constructor.

For Java func^s use Javadocs oracle platform

- tostring() → It converts your object into

- scanner → built-in class which we will use for taking i/p from user.

syntax → scanner sc = new scanner (system.in);
sc # int n = sc.nextInt();
string s = sc.next();

Function (extra class) 16/03/22.

literal, operator, data type
variable - size of variable, datatype of variable.

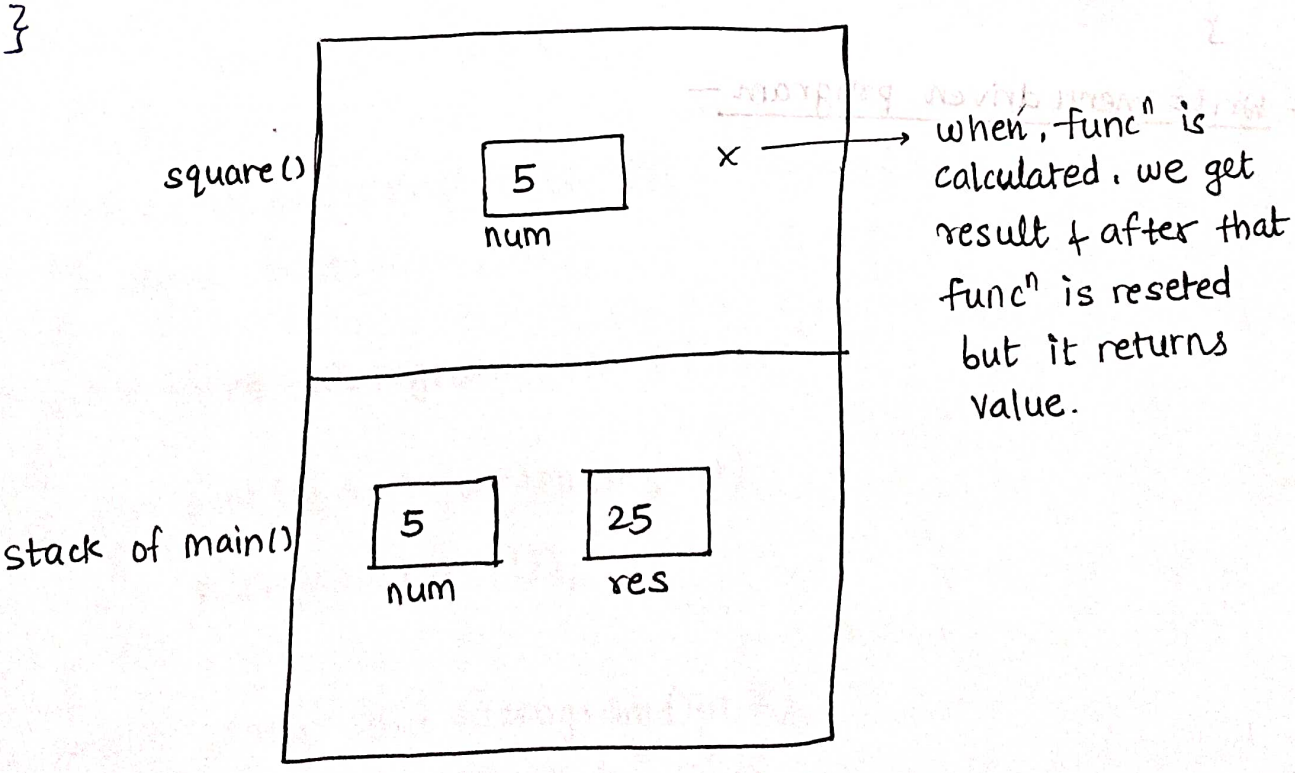
POG - Pointer, Array, Array with pointer.
Pointer pass into function as parameter.
Pointer returning from function.

Functions —

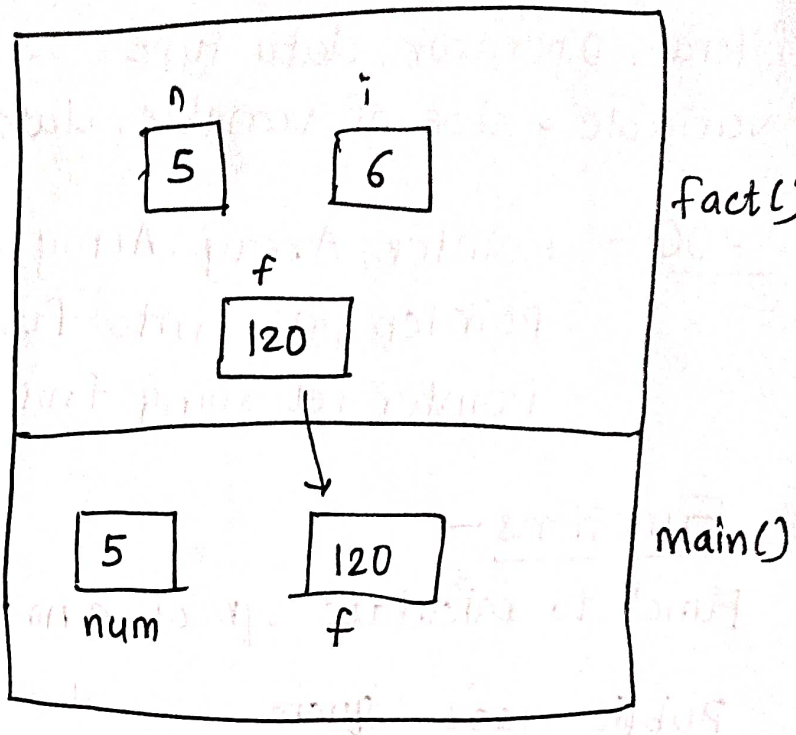
Func^n to calculate sqr of a no.

```
Public class square
{
    public static void main (string [] args)
    {
        int num, res;
        num = 5;
        res = square(num);   // num is
                                actual parameter
    }
    static int square (int n) // n is formal parameter
    {
        return n*n;
    }
}
```

Formal parameter initialize with normal parameter.



```
static ~~fact~~ int fact (int n) // n=5
{
    int i, f = 1;
    for (i=1, i<=n; i++) //i=6
    {
        f = f * i;    // f=120
    }
    return f;
}
```



Sum of digits —

```
int sum_of_digits (2543/int n)
{
    int rem, sum = 0;
    while (n>0)
    {
        rem = n % 10;
        sum = sum + rem;
        ~~rem~~ = n = n/10;
    }
    return sum;
}
```

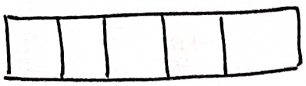
Write menu driven program —

Array.

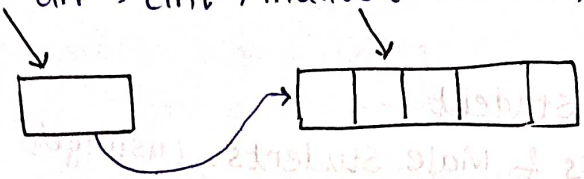
17/03/22.

C

int arr[5]



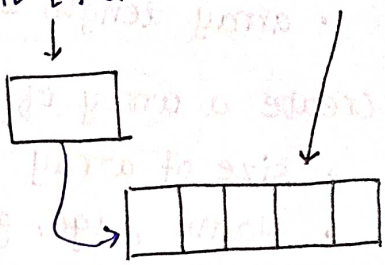
int *arr = (int*) malloc (sizeof(int)*5);



- object is created on stack & heap also.

Java

int [] arr = new int [5];



- object is created on heap.

```
import java.util.*;
pubic class Main
{
    public static void main (String [] args)
    {
        int [] arr = new int [5] ;

        Scanner sc = new Scanner ( System.in)
        System.out.println (" Enter 5 no.s:" );

        for ( int i=0 ; i< arr.length ; i++)
        {
            arr [i] = sc.nextInt();
        }

    // System.out.println (" arr [3] = " + arr [3]);
        print_histogram (arr);
    }
}
public static void print_histogram (int [] arr )
{
    for (int i =0; i< arr.length ; i++)
    {
        histogram (arr[i]);
    }
}
public static void histogram (int x)
{ System.out.println ( x + " : ");
  for (int i =0; i<~~arr.length~~ x ; i++ )
  { S.O.P ("*");} } }
```

gives length of array.

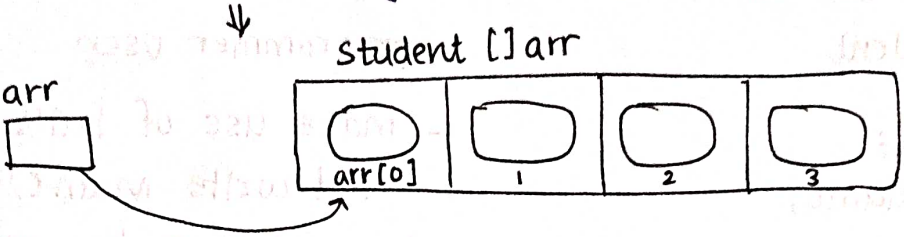
arr.length

Array with diff. datatypes

```
Student [] arr = new Student [4];

arr [0] = new Student ("Ram", 18, 45);
arr [3] = new Student ("Shyam", 20, 56);
```

} for user-defined data-types.




```
char[]arr = new char[7];
int [] arr = new ~~char~~ int [5];
float [] arr = new float [10];
```

} for built-in datatypes.

to take all i/ps

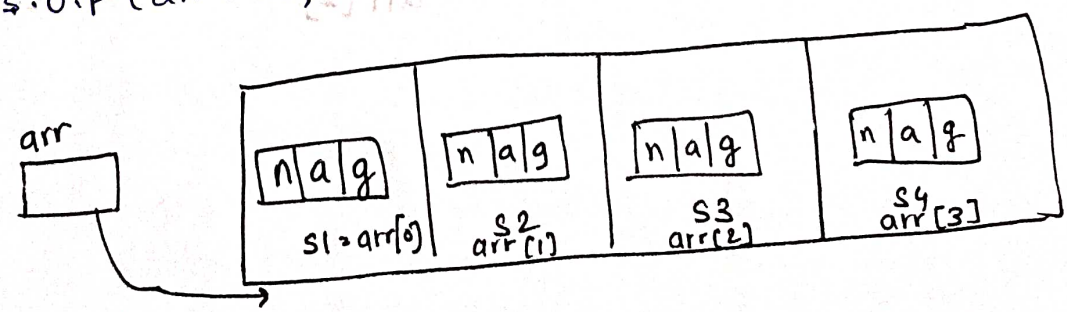
```
Student [] arr = new Student [5];
for (int i=0 ; i< arr.length ;i++)
{
    scanner sc=new scanner (System.in);
    ~~String int~~ n =sc.next ();
    S.o.p (" Enter student name: ");
    String n = sc.next();
    S.o.p (" Enter student age :");
    int a = sc.nextInt ();
    S.o.p (" Enter student gender:");
    char g = sc.next().charAt(0);

    arr[i] = new student (n, a, g);
}
```

Array
- datatypes
- with objs.
- this
- private func^n (make)

for all Female students

```
for (i=0 ; )
{ if (arr[i].getGender () == 'F')
    sys.o.p (arr [i]);
}
```

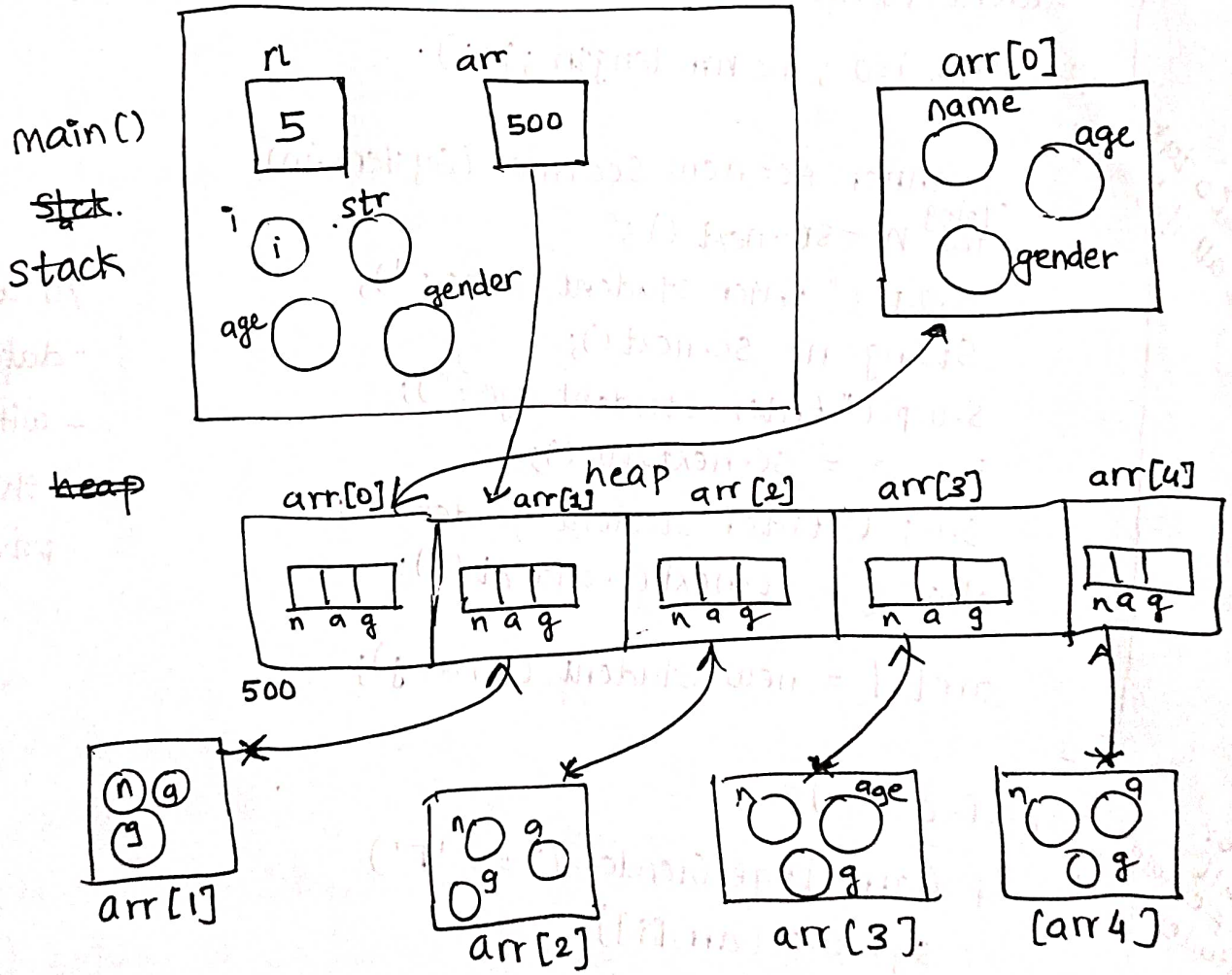


** Student class Program -

- developer user - who decides the classes (entities & classify those entities).
  - write different classes.

```
class Student
{
    int ron;
    String name;
    int age;
}
    Constructor
    setter & getter
    toString
}
```

- programmer user
  - make use of built-in classes and write main()
  - use of class to create object
  - call functions using those object.



19-3-22

Q. What supports java (pass by value or pass by reference)

Java supports pass by value only.
↳ new reference is created but is pass by value.

Command Line Argument -

To convert string into different data types —

string to data type conversion

```
int num1 = Integer.pargeInt (args[0]);
float f1 = Float.parseFloat ("34.56");
boolean b = Boolean.parseBoolean ("true");
```

↳ command line argument take i/p as a string -
↳ Bydefault → string ⟹ then convert to desired data type.

↳ • takes i/p at time of run, cmdLine Arg.
• other takes i/p after running (i.e. holds the i/p).

e.g. for student Class -

```
public class Main {
    p s v m (string [] args) {
    • Student s1 = new student (args[0], Integer.parseInt (args[1]),
                                arg[2].charAt [0]);

      or
    • int a = Integer.parseInt (args[1]);
      char g = args[2].charAt[0];
      student s1 = new student (args[0], a, g);
    • sysout (s1);
```

↳ Run → java Main Ram, 20 male

Ans — Ram, 20, m

Final Keyword

in C → const
(we can change it using pointer)
(value is not 100% fix)

in Java → final
(we cannot change it)
bcz we cannot use pointers in Java.
(100% fix).

e.g.
```
final float pi=3.14f
```

Static Variable →

Single copy of this variable is maintained throughout prgm. i.e. sin for all objects.

e.g. Static

works on → / access ↓	class object level: static func^n	Object level: non-static f^n
static variable	Yes ✓	Yes ✓
Non-static variable	No X	Yes ✓

- **Menu Driven Program-**

- we write func^n to perform a task.
- The func^n should return a value (result), it should not print the result inside it.
- It is written for printing then return datatype should be void otherwise not.

Q. Write a menu driven program to print following Menu -

① Cold drink - (f) q+r
- i. Mirinda - 22 ₹ /-
- ii. Coke → 25/-
- iii. Pepsi → 20/-
- iv. Cold coffee → 50/-
- v. Exit

② Hot drink - (f) q+r
- i. Tea → 15/-
- ii. Coffee → 30/-
- iii. Hot Chocolate → 45/-
- iv. Bournvita → 20/-
- v. Exit

③ Snack - (f) q+r
- ① Vada pav → 25/-
- ② Dabeli → 20/-
- ③ Samosa → 18/-
- ④ Noodles → 30/-
- ⑤ Manchurian → 35/-
- ⑥ Exit.

④ Ice-creams - (f) q+r
- ① Butterscotch → 35/-
- ② Black Current → 37/-
- ③ Mango → 25/-
- ④ Vanilla → 20/-
- ⑤ Almond → 50/-
- ⑥ Exit.

Total static. S.o.p (Total Bill Amount:);