Kunal Maheshwari
B11CSE012

Q.1

class Intstack.h file

```
#inndef INSTACK_H // prevent multiple
#define INSTACK_H ;

class Intstack
{
 Public:
    Intstack();
    Intstack(const Intstack&);
    void reset();
    void push(int n);
    int pop();
    bool is_full();
    bool is_empty();
   ~Intstack(); //required (heap used)
 private:
    int * arr;
    int top;
    int size; //added additional.
};

#endif
```

//top always at arr[0].

Kunal Maraskar
[illegible]
Page No. [illegible]
Date

Intstack.cpp

```
# include <iostream>
# include "Intstack.h"
using namespace std;

Intstack :: Intstack()   // constructor
{
    this->arr = new int [100]; //array of 100 ints.
    this->top = -1 ;
    this->size = 0;
}

Intstack :: Intstack (const Intstack &s)
{
    this->arr = new int [100] ; // allocate new
    this->top = s.top ;          memory
    this->size = s.size ;        explicitly.
    for(int i=0 ; i<100 ; i++)
    {
        this->arr[i] = s.arr[i] ;
    }
}

                         Intstack :: ~Intstack()
void Intstack :: Push(int n) } {
{                                delete [] this->arr;
                                 // free heap memory.

                                 }
}
```

```
void IntStack :: reset ()
{ this->size = 0;
  this->top = -1;
}
bool IntStack :: is_full ()
{
   if (this->size == 100)
   { return true; }
  return false;
}

bool IntStack :: is_empty()
{
    if (this->size == 0)
    { return true; }
   return false;
}

void IntStack :: push(int n)
{  if( ! this->is_full() ) // if stack is not full
   { if (this->is_empty()) // if stack is empty.
     { this->top = 0;
       this->arr[top] = n;
     } this->size++; // increase size;
     else
     {
       for(int i=0; i<size; i++)
       for( int i=size-1; i>=0; i--) //shift elements
       {
          this->arr[i+1] = this->arr[i];
       }
       this->arr[0] = n;
       this->size++; // inc size;
     }
}  }
```

Kunal Moharkar
BT18CSE018

Page No.
Date:

```
int Instack :: Pop()
{   int retval = -1 // if pop fails;
    if ( ! this -> is_empty()) // stack not empty.
    {
        int retval = this -> arr [top];

        for(int i=0; i< size-1; i++ ) //shift all
        {
          this -> arr[i] = this -> arr [i+1]

        }
        this -> size-- ; // decrease size

        if (this -> is_empty ()) //case for empty
        {
            this -> top = -1;
        }
    }

    return retval;
}
```

(b) overload push

```
void IntStack :: Push ( int a[], int array_size)
{
    // repeated calls to push method.
```

~~Kunal~~ Kunal mohrekar
BT18CSE018
Page No: /Snehanka.
Date.

```
for(int i=0; i<array_size; i++)
{
    if(!this->is_full())  //stack not full
    {
        this->push(a[i]);
    }

}
}
```

(c) overload pop()

```
void IntStack :: pop(int a[], int n)
{   // multiple pop calls.
    for(int i=0; i<n; i++)
    {
        if(!this->is_empty())  //stack not empty
        {
            ~~this->pop()~~
            a[i] = this->pop();
        }

    }
}
```

Kunal Mohorkar
BT18CSE018

(d) Modifying class defination: we need to add a print method in class stack. which should be its friend:

~~class stack~~
~~{~~
~~private~~:

```
class intstack
{
  Public :
    friend void print_stack (Intstack * pstack);
};
```

main.cpp

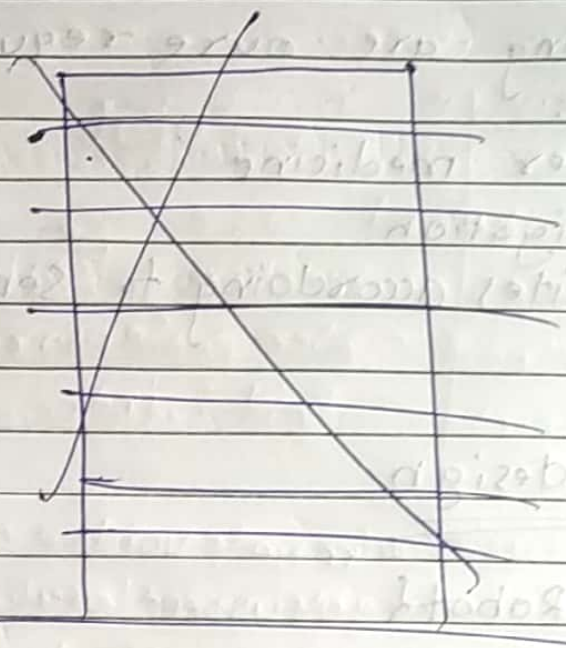
```
int main()
{
  void print_stack ( Intstack * pstack )
  {
    for(int i=0 ; i < Pstack->size ; i++)
    {
      cout << pstack->arr[i] << "\n";
    }
  }
}
```

~~direct access~~ as friend is present

Kunal Mohankar
B718CSE018

Page No.
Date.

Q.4 UML Diagram.



class name: IntStack

data members:
- arr : int *
- top : int
- size : int (added)

functions:
+ IntStack()
+ ~IntStack()
+ IntStack(IntStack : const &)
+ Push(n : int) : void
+ Pop() : int
+ is_full() : bool
+ is_empty() : bool
+ reset() : void
+ Pop(a : int[], n : int) : void
+ Push(a : int[], arrsize : int) : void

Kunal Moharkar
B718CSE018

Q.3

Following are ~~our~~ requirements for class robot:

1. ~~Deliver~~ medicine
2. navigation
3. activites accordaing to sensors

Class design

```
Class Robot{

Private:
    Position position; // position.
    Sensor   sen;  // sensor data
    int      R_id; // id of robot

};

struct position{
    int longitude  // exact position.
    int latitude
};

struct Sensor{  // sensor data.
    int stopping_dis;
    int photo sensors;
}
```

Kunal Moharkar
13T18CSE018

Page No.
Date. Homework.

Public:

void move (Position destination); //sensor data also updates.
// move to appropriate location

Position getCurrentLocation();
// returns the current location

bool will_stop(); //offset detection.
// tells whether robo will stop or not.

void stop();
//stop the robot

void PickMedicine();
// picks the required medicines;

void DropMedicine();

// drops the medicines at the required location;

void connectPhotoRgister()
// connects to the photoregisters.

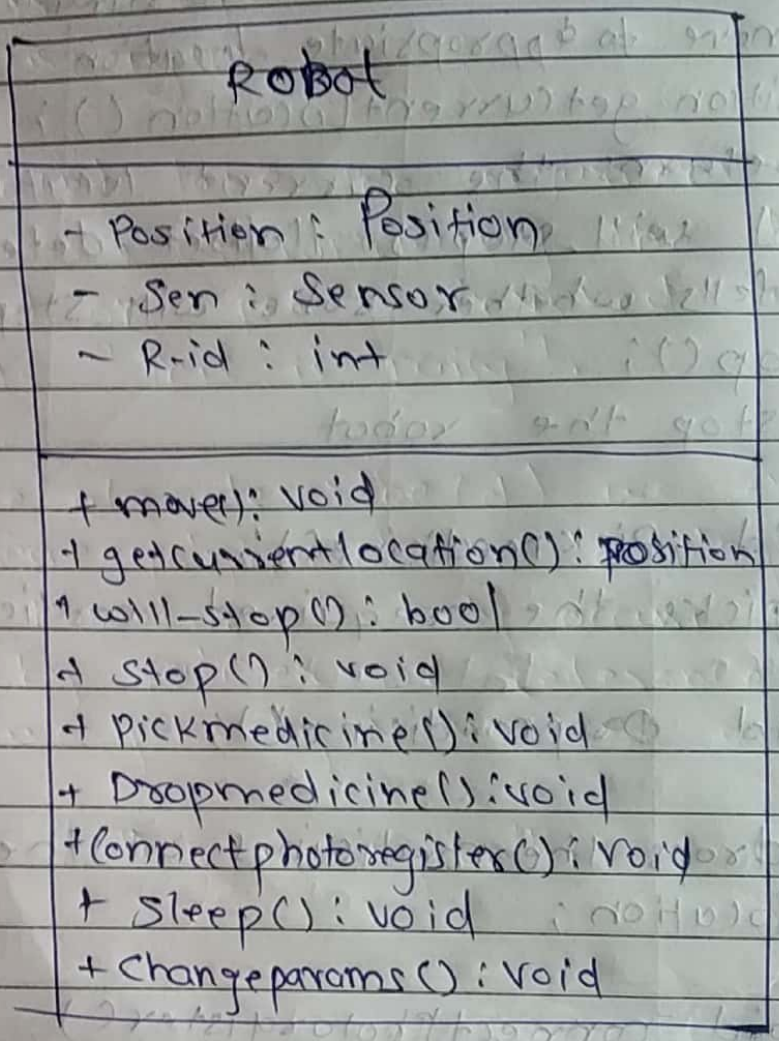
void Sleep();
// turns of robot.

void changeparams();
// modify sensor data.

Kunal Moharkar
BT18CSE018

Page No:
Date:

UML



Kunal Mohorkar
BT18CSE018

Page No.
Date.

Q.2

a) ~~In this code~~

Output:

Base Constructor
Derived Constructor
Base Destructor

Yes, the memory resource leak occurs in this case as obj of type base class so when memory is freed using delete keyword only the datamembers of base class will be freed but those of derived will remain as a garbage.

Solution →

make destructor of base class virtual:

virtual ~a() { cout << " Base Destructor " << "\n" }

the above is a general (should be done)

However, as there are not datamembers present here in derived class no such resource leak should occur.

Kunal Mohankar
BT18CSE018

Page No:
Date:

(b) Output:
This is Daisy.
This is Tyson

describe(p) take the argument pet by value. So appropriately the description method on pet is invoked and outputs "This is Daisy".

In case of describe(d) we expect run time polymorphism to play its part However it only occurs on pass by pointer or reference but here we pass by value. So again

O/P → "This is Tyson".