SINHGAD COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

SUBJECT CODE: 310248

LAB MANUAL
Laboratory Practice-I

Semester – I, Academic Year: 2022-23

Third Year of Computer Engineering (2019 Course)
310248: Laboratory Practice I

Teaching Scheme Practical: 04 Hours/Week	Credit:02	Exam Scheme & Marks Term work: 25 Marks Practical: 25 Marks

Course Objectives:

- To learn system programming tools
- To learn modern operating system
- To learn various techniques, tools, applications in IoT and Embedded Systems /Human Computer Interface/Distributed Systems/ Software Project Management

Course Outcomes: On completion of the course, learners will be able to

- Systems Programming and Operating System

CO1: Implement language translators
CO2: Use tools like LEX and YACC
CO3: Implement internals and functionalities of Operating System

CO-PO Mapping Matrix												
CO/PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO 1	1	2	2	2	3	-	-	-	-	-	-	1
CO 2	1	2	2	2	2	-	-	-	-	-	-	1
CO 3	1	2	2	2	2	-	-	-	-	-	-	1
CO 4	1	2	3	2	-	2	-	-	2	1	2	-
CO 5	1	2	2	1	-	2	-	-	3	2	1	-
CO 6	2	2	2	1	-	2	-	-	2	-	2	1

List of Laboratory Assignments

Prof. Supriya Patil
Prof. Ankita Kotalwar
Prof. Laxman Pawar
Lab Coordinator

Dr. M.P. Wankhade
HoD, Dept of Computer Engg

Instructions:

Guidelines for Instructor's Manual

The instructor‘s manual is to be developed as a reference and hands-on resource. It should include prologue (about University/program/ institute/ department/foreword/ preface), curriculum of the course, conduction and Assessment guidelines, topics under consideration, concept, objectives, outcomes, set of typical applications/assignments/ guidelines, and references.

Guidelines for Student's Laboratory Journal

The laboratory assignments are to be submitted by student in the form of journal. Journal consists of Certificate, table of contents, and handwritten write-up of each assignment (Title, Date of Completion, Objectives, Problem Statement, Software and Hardware requirements, Assessment grade/marks and assessor's sign, Theory- Concept in brief, algorithm, flowchart, test cases, Test Data Set(if applicable), mathematical model (if applicable), conclusion/analysis. Program codes with sample output of all performed assignments are to be submitted as softcopy. As a conscious effort and little contribution towards Green IT and environment awareness, attaching printed papers as part of write-ups and program listing to journal must be avoided. Use of DVD containing students programs maintained by Laboratory In-charge is highly encouraged. For reference one or two journals may be maintained with program prints in the Laboratory.

Guidelines for Laboratory /Term Work Assessment

Continuous assessment of laboratory work should be based on overall performance of Laboratory assignments by a student. Each Laboratory assignment assessment will assign grade/marks based on parameters, such as timely completion, performance, innovation, efficient codes, punctuality and Guidelines for Practical Examination

Problem statements must be decided jointly by the internal examiner and external examiner. During practical assessment, maximum weightage should be given to satisfactory implementation of the problem statement. Relevant questions may be asked at the time of evaluation to test the student’s understanding of the fundamentals, effective and efficient implementation. This will encourage, transparent evaluation and fair approach, and hence will not create any uncertainty or doubt in the minds of the students. So, adhering to these principles will consummate our team efforts to the promising start of student's academics.

Guidelines for Laboratory Conduction

The instructor is expected to frame the assignments by understanding the prerequisites, technological aspects, utility and recent trends related to the topic. The assignment framing policy need to address the average students and inclusive of an element to attract and promote the intelligent students. Use of open source software is encouraged. Based on the concepts learned. Instructor may also set one assignment or mini-project that is suitable to respective branch beyond the scope of syllabus. For the elective subjects students should form group of 3-4 students. The faculty coordinator will take care that all the assignment should be assigned to class and minimum two assignments are compulsory for each group.

Programming tools recommended: - Human computer Interface-GUI in python

Internet of Things and Embedded System- Raspberry Pi/Arduino Programming; Arduino IDE/Python Interfacing. Other IoT devices

Software project management-MS project/Gantt Project/Primavera

Virtual Laboratory:

- http://cse18- iiith.vlabs.ac.in/Introduction.html?domain=Computer%20Scie nce
- http://vlabs.iitb.ac.in/vlabs-dev/labs/cglab/index.php

Course Objectives:

- To learn system programming tools.
- To learn modern operating system

Course Outcome:

On completion of this course, learners will be able to

- CO1: Implement different system software's like assembler, macro processor, DLL, etc.
- CO2: Implement concept of synchronization and concurrency.
- CO3: Implement scheduling policies and memory management concepts of operating system

Assignment No.: 01

Problem Statement: Design suitable Data structures and implement Pass-I and Pass-II of a two-pass assembler for pseudo-machine. Implementation should consist of a few instructions from each category and few assembler directives. The output of Pass-I (intermediate code file and symbol table) should be input for Pass-II.

Objectives:

1. To study the design and implementation of pass I and pass II of two pass assembler.
2. To study the categorized instruction set of assembler.
3. To study the data structure used in assembler implementation.

Theory:

Assembler is a program for converting instructions written in low-level assembly code into relocatable machine code and generating along information for the loader.

Design of two Pass Assemblers

An assembler is a translator, that translates an assembler program into a conventional machine language program. Basically, the assembler goes through the program one line at a time and generates machine code for that instruction.

What Is A Single Pass Assembler?

It is an assembler that generally generates the object code directly in memory for immediate execution. It parses through your source code only once and you are done.

Why Do We Need A Two-Pass Assembler?

As explained, the one-pass assembler cannot resolve forward references of data symbols. It requires all data symbols to be defined prior to being used. A two-pass assembler solves this dilemma by devoting one pass to exclusively resolve all (data/label) forward references and then generate object code with no hassles in the next pass. If a data symbol depends on another and this another depends on yet another, the assembler resolved this recursively.

Advantages of Two Pass Assembler

One of the main advantages of Two-Pass Assembler is that many times the first pass of an extreme Two-pass assembler generates the output file which is then read by the second pass.

The advantage of this is that first pass can record each line of input, along with that the next position of some or all lexemes of that line and some of the results of parsing.

For example, in an assembly-level language, each line can be sent to the other line by a record indicating if the line was a definition or a statement and if statement then if it contained a label and what was the value of the location counter was before processing that line.

The use of such kind of intermediate files can eliminate unnecessary computations in the two-pass assembler but then it adds to the input-output burden of the system.

It generates instructions by evaluating the mnemonics (symbols) in operation field and find the value of symbol and literals to produce machine code. Now, if assembler do all this work in one scan then it is called single pass assembler, otherwise if it does in multiple scans then called multiple pass assembler. Here assembler divide these tasks in two passes:

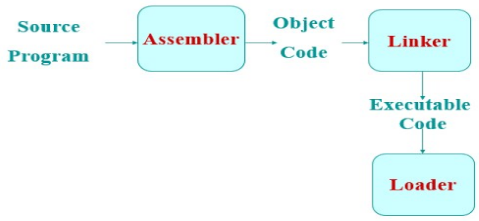
- **Pass-1:**
    1. Define symbols and literals and remember them in symbol table and literal table respectively.
    2. Keep track of location counter
    3. Process pseudo-operations
- **Pass-2:**
    1. Generate object code by converting symbolic op-code into respective numeric op-code
    2. Generate data for literals and look for values of symbols

Pass I of the Assembler
Data Structure used in Pass I

1. OPTA
2. SYMTAB
3. LITTAB
4. POOLTAB

Algorithm
Intermediate code
Declaration and Assembler Directive Processing

Pass II of the Assembler



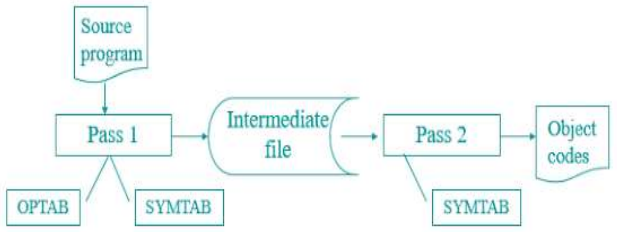
Introduction

- Convert mnemonic operation codes to their machine language equivalents
- Convert symbolic operands to their equivalent machine addresses
- Build the machine instructions in the proper format
- Convert the data constants to internal machine representations

- Write the object program and the assembly listing

Two Pass Assembler

- *Read from input line*
  - LABEL, OPCODE, OPERAND



Algorithm/Flowchart:
Pass 1of Assembler:
Step 1:

- Prepare and initialize all data structures like MOT, POT, ST etc.
- Open input soured file and output intermediate file in respective mode.
- Initialize all pointers(i.e. MOTPTR, POTPTR etc.) to point to the first entry.
- Initialize the LC to φ.
- Set error flag = OFF

Step 2:

- Read a line of source code.
- If 'end of file' OR 'END' pseudo opcode
- If error flag = off
- Go to pass 2.
- Else disp_message "Unsuccessful assembly" and do the reverse of 1st step and quit.
- Else
- Goto step 3

Step 3:
Analyze the statement
Case 1: Statement contains only comment.

•
Start form SOC i.e. start of comment and ignore everything till EOC
i.e. end of comment

•
GOTO STEP 2
Case 2: Statement contains only label (e.g. loop:)

- Check whether label is present in the ST. (i.e. a symbol with attr. Label)
If not present Insert label in ST & also mark attribute as ‘L’.
address of label is set in LC.
insert this address against label in ST.
else error “duplicate label”. And
set error flag = ON
- GOTO STEP 2

Case 3:
Statement contains only mnemonic (e.g. STOP)

- Search MOT for Mnemonic
- LC = LC + length of instruction
- GOTO STEP 2

Case 4:
Statement contains mnemonic and operand. (e.g. LOAD A)

- Search MOT for mnemonic
- Search operand in ST if present do nothing.
else insert operand in ST
- LC = LC + length of instruction
- GOTO STEP 2

Case 5:
Statement contains label and mnemonic (e.g. back : STOP)

- Check whether label is present in ST
if not present
insert label in ST & attribute as ‘L’
address of label is set to LC
insert this address against label in ST
else error ”duplicate label ”. And set error flag = ON
- Search MOT for mnemonic
- LC = LC + length of instruction
- GOTO STEP 2

Case 6 :
Statement contains label, mnemonic and operand (e.g.LOOP: jmp loop)

- Check whether label is present in ST
if not present insert label in ST & attribute as ‘L’
address of label is set to LC
insert this address against label in ST

else error ”duplicate label ”. And set error flag = ON

- Search MOT for mnemonic
- Search operand in ST
If present do nothing else insert operand in ST
- LC = LC + length of instruction
- GOTO STEP 2

Case 7:
Statement contains pseudo opcode
Case 7.1: Statement contains pseudo opcode ‘START ’ or ‘ORG’ (e.g. ORG 1φφφ)

- Search POT for pseudo opcode
- Read the next operand and initialize LC with the value of the operand
- GOTO STEP 2

Case 7.2: Statement contains pseudo code ‘ENDP’ (e.g. ENDP)

- Search POT for pseudo code
- Reset code segment flag and set data segment flag
.
- GOTO STEP 2

Case 7.3: Statement contains pseudo code ‘dB’ (e.g. dB SUM 5)

- Search POT for pseudo code
- Search operand in the ST
-
put the address against operand in ST.
-
put next operand (e.g. in this case 5) against address i.e. put value (if any) against the address.
-
LC = LC + 1 as 1 is length of db.
- GOTO STEP 2

Case 7.4: Statement contains pseudo code ‘dw’ (e.g. dw SUM ?)

- Search POT for pseudo code
- Search operand in the ST
put the address against operand in ST.
read next operand
if then do LC = LC + 2
else
1) put next operand against the address
2) LC = LC + 2

•
GOTO STEP 2
Case 7.5 : Statement contains pseudo code ‘CONST’ (e.g. ONE CONST
•
Search POT for pseudo code
•
Search operand in ST
put the address against operand in ST
read the next operand
1) put value of the next operand against the address (e.g. in this case 1)
2) LC = LC +1
•
GOTO STEP 2
Case 7.6 : Statement contains pseudo code ‘END’ or ‘END OF FILE’
•
Search POT for pseudo code
•
If error flag = ON don’t goto
pass 2, else GOTO pass 2
Case 8 : Literal Handling
•
If the literal is found as a part of instruction then, put it in ST with appropriate attribute & value if such a kind of literal is not already there.
•
Else write the literal name, literal value and present LC where literal is occurring, in the ST (Symbol Table)
Case 9 : Procedure Handling
Case 9.1 : Procedure definition start
IF FIRST TOKEN is PROC[if definition flag for procedure is already set, then it is an instance of definition inside definition, which is not allowed, so display error “nested definition procedure not allowed “ else go ahead]
-
Read the next token which is the name of
the procedure
-
Validate with st
-
If already present as a definition
,
Display error : Duplicate procedure definition” and set error flag on ELSE
-
put the name and corresponding attributes along with present LC value as the starting
address of the procedure in St
-
set definition flag
-
GOTO STEP 2
Case 9.2 : Procedure definition ends
-
reset the respective procedure definition flag to write total length taken by procedure, (if
reqd) in ST

GOTO STEP 2
Case 9.3 : Procedure call
-
First token must be 'CALL' or something like, so validate
through MOT
-
Read the next token
which must be name procedure
validate that name for the correct attributes, if valid, do
nothing else Display_error("problem in the procedure call")
and set error flag pn
-
GOTO STEP 2
Case 10 : Default
An assembler comes to this state only if the instruction is invalid i.e it is not starting with label, mnemonic or pseudo opcode or comment
-
ignore the line
-
set error flag ON
-
GOTO STEP 2

Pass 2 of an assembler:
Step 1:
1)
Assuming all tables are initialized in pass 1, initialize table pointers.
2)
Open input source file and output target file in respective mode.
3)
Initialize all pointers (i.e. MOTPTR, POTPTR etc.) to point to the first entry.
4)
Initialize the LC to φ.
Step 2:
Read a line of source code
If 'end of file' or 'END' pseudo opcode
Assembly complete. Reset all tables, variables and close all files.
else goto step 3.
Step 3:
Analyze the statement
Case 1: Statement contains only comment.
•
Start form SOC i.e. start of comment and ignore everything till EOC
i.e. end of comment
•
GOTO STEP 2
Case 2:
Statement contains only label (e.g. loop:)
•
Ignore this as pass 1 has taken care of label definition handling
•

GOTO STEP 2
Case 3:
Statement contains only mnemonic (e.g. STOP)
- Search MOT for mnemonic
- Write opcode for that mnemonic in output file
- LC = LC + length of instruction
- GOTO STEP 2

Case 4:
Statement contains mnemonic and operand (e.g. LOAD A)
- Search MOT for mnemonic
- Write opcode for that mnemonic in output file
- Search operand in ST
if present , get address of symbol from ST and write in output file
- LC = LC + length of instruction
- GOTO STEP 2

Case 5:
Statement contains label and mnemonic (e.g. back: STOP)
- Ignore label as pass 1
has tackled it.
- Search MOT for mnemonic
- Write opcode for that mnemonic in output file
- LC = LC + length of instruction
- GOTO STEP 2

Case 6:
Statement contains label, mnemonic and operand (e.g. LOOP : jmp
loop)
- Ignore label as pass 1 has taken care of it.
- Search MOT for mnemonic
- Write opcode for that mnemonic in output file
- Search operand in ST
if present , write address of symbol in output file
- LC = LC + length of instruction

- GOTO STEP 2

Case 7:
Statement contains pseudo opcode
Case 7.1: Statement contains pseudo opcode 'START' or 'ORG' (e.g. ORG1 φφφ)

- Search POT for pseudo opcode
- Read the next operand and initialize LC with the value of the operand
- GOTO STEP 2

Case 7.2 : Statement contains pseudo code 'ENDP' (e.g. ENDP)

- Search POT for pseudo code
- Reset code segment flag and set data segment flag.
- GOTO STEP 2

Case 7.3: Statement contains pseudo code 'DB' (e.g. DB SUM 5)

- Search POT for pseudo code
- Search operand in the ST
- put next operand value (if any) against the present LC
- write this addr. In output file
- LC = LC + 1
- GOTO STEP 2

Case 7.4: Statement contains pseudo code 'DW' (e.g. DW SUM ?)

- Search POT for pseudo code
- Read the next operand and search it in ST
if then write some garbage value or some predefined value against
the present LC and also in output file
- LC = LC + 2
- GOTO STEP 2

Case 7.5 : Statement contains pseudo code 'CONST' (e.g. ONE CONST 1)

- Search POT for pseudo code
- Search operand in ST
1) put value of the next operand against the present LC (e.g. in this case 1)
2) LC = LC +1
3) write this constant value in output file
-

GOTO STEP 2
Case 7.6 : Statement contains pseudo code 'END'
•
Search POT for pseudo code
•
End of assembly

Input:

Source code of Assembly Language

```
SAMPLE START 100
        USING *, 15
        L 1, FOUR
        A 1, =F'3'
        ST 1, RESULT
        SR 1, 2
        LTORG L 2,
        FIVE
        A 2, =F'5'
        A 2, =F'7'
FIVE DC F'5'
FOUR DC F'4'
RESULT DS 1F
END
```

Output:

```
100 SAMPLE START 100
100 USING *, 15
100 L 1, FOUR
104 A 1, =F'3'
108 ST 1, RESULT
112 SR 1, 2
114 LTORG
124 L 2, FIVE
128 A 2, =F'5'
132 A 2, =F'7'
136 FIVE DC F'5'
140 FOUR DC F'4'
144 RESULT DS 1F
152 5
156 7
160 END
```

Machine Opcode Table (MOT)

Mnemonic	Hex / Binary Code	Length (Bytes)	Format

L	5A	4	RX
A	1B	4	RX
ST	50	4	RX
SR	18	2	RR

Pseudo Opcode Table (POT)

Pseudo op	Address / Name of Procedure to implement pseudo operation
START	PSTART
USING	PUSING
DC	PDC
DS	PDS
LTORG	PLTORG
END	PEND

Symbol Table (ST)

Sr. No	Symbol name	Address	Value	Length	Relocation
1	SAMPLE	100	--	160	R
2	FIVE	136	5	4	R
3	FOUR	140	4	4	R
4	RESULT	144		4	R

Literal Table (LT)

Sr. No	Literal	Address	Length
1	3	120	4
2	5	152	4
3	7	156	4

Base Table (BT)

Register no Availability Value/ Contents1 N

--
: : :
: : :
: : :
15 Y 100

Object Code

LC OPCODE OPERAND

---

100 5A 1,40(0,15)
104 1B 1,20(0,15)
108 50 1,44(0,15)
112 18 1,2
124 5A 2,36(0,15)
128 1B 2,52(0,15)
132 1B 2,56(0,15)

Software Requirement:

1. Fedora
2. Eclipse
3. JDK

Hardware Requirement:

Not specific

Conclusion:

Input assembly language program is processed by applying Pass-I and Pass-II algorithm of assembler and intermediate data structures, Symbol Table, Literal Table, MOT, POT, BT, etc. are generated.

Assignment No.: 02

Problem Statement: Design suitable data structures and implement Pass-I and Pass-II of a two-pass macro- processor. The output of Pass-I (MNT, MDT and intermediate code file without any macro definitions) should be input for Pass-II.

Objectives:

1. To identify and design different data structure used in macro-processor implementation.
2. To apply knowledge in implementation of two pass microprocessor.

Theory:

A macro definition is a way to give a name to a piece of text.

• After a macro has been defined, the programmer can write the macro name instead of the piece of program.

• A macro is, in effect, an abbreviation for a piece of text.

A Macro instruction is the notational convenience for the programmer. For every occurrence of macro the whole macro body or macro block of statements gets expanded in the main source code. Thus Macro instructions make writing code more convenient.

Salient features of Macro Processor:

- **Macro** represents a group of commonly used statements in the source programming language.
- Macro Processor replaces each macro instruction with the corresponding group of source language statements. This is known as the expansion of macros.
- Using Macro instructions programmer can leave the mechanical details to be handled by the macro processor.
- Macro Processor designs are not directly related to the computer architecture on which it runs.
- Macro Processor involves definition, invocation, and expansion.

Pass 1 (Definition of MACROS)

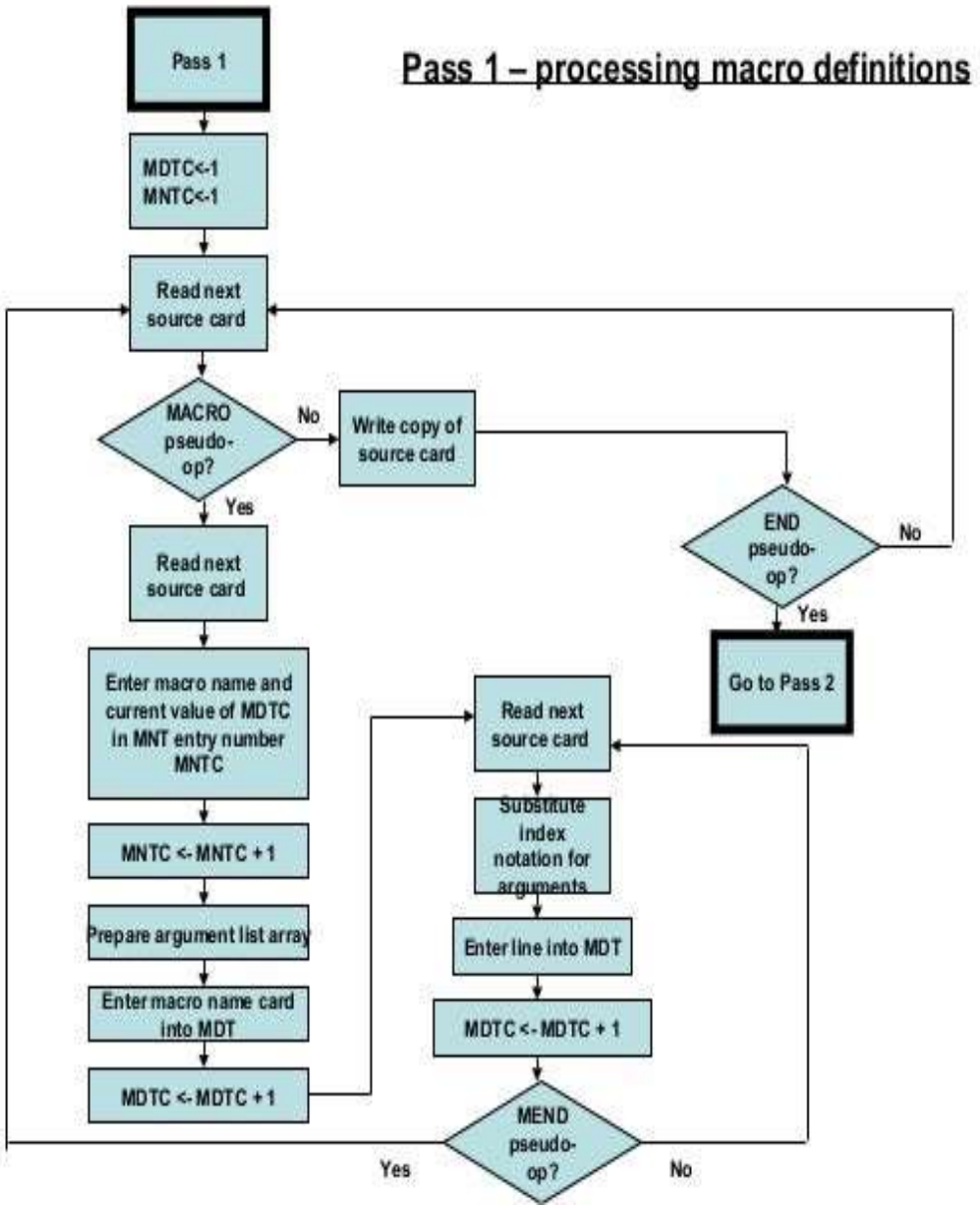
- Pass1 of macro processor makes a line-by-line scan over it's input.
- Set MDTC = 1 and MNTC = 1.
- Read next line of input program.
- If it is a MACRO pseudo-op, the entire macro definition except MACRO line is stored in MDT.
- The name is entered in the MNT along with a pointer to the 1st location of MDT entry.
- When the END pseudo-op is encountered all the macro-definitions have been processed, so control is transferred to pass2.

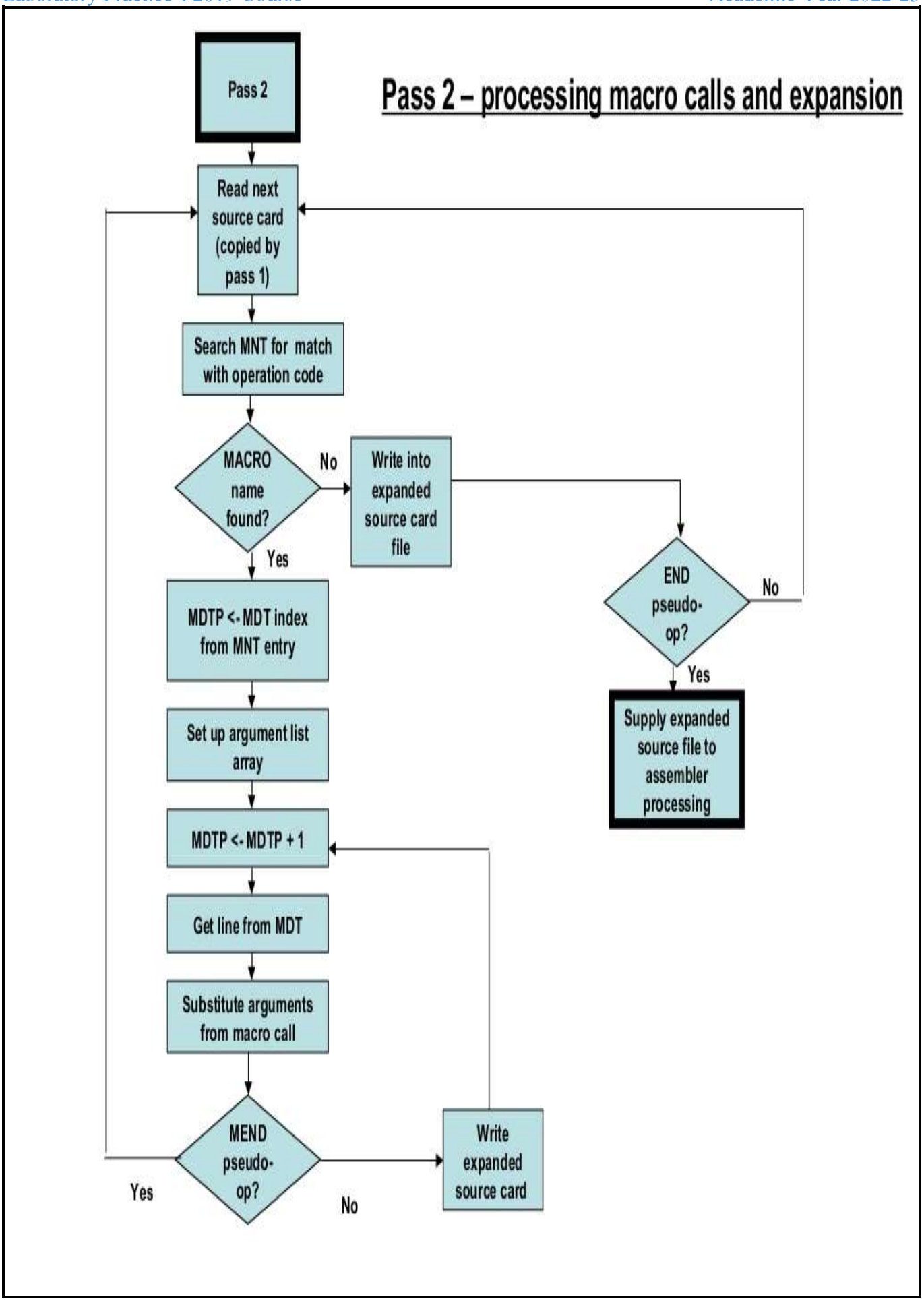
Pass 2 (Replacing MACRO calls by its definition)

- This algorithm reads one line of i/p program at a time.
- For each line it checks if op-code of that line matches any of the MNT entry.
- The initial value of MDTP is obtained from MDT index field of MNT entry.

- The macro expander prepares the ALA consisting of a table of dummy argument indices and corresponding arguments to the call.
- The value from the argument list is substituted for dummy arguments indices in the macro definition table.
- Reading MEND line in MDT terminates expantion of macro and scanning continues in the input file.
- When END pseudo-op is encountered, the expanded source program is given to the assembler.

Algorithm/Flowchart:




Pass 2
Pass 2 – processing macro calls and expansion
Read next source card (copied by pass 1)
Search MNT for match with operation code
MACRO name found?
No
Write into expanded source card file
Yes
END pseudo-op?
No
Yes
Supply expanded source file to assembler processing
MDTP <- MDT index from MNT entry
Set up argument list array
MDTP <- MDTP + 1
Get line from MDT
Substitute arguments from macro call
MEND pseudo-op?
Yes
No
Write expanded source card

Input:

Pass1 of 2 Pass Macro Processor

Source Program

MACRO
&LAB INCR &ARG1, &ARG2,&ARG3
&LAB ADD AREG, &ARG1
ADD AREG, &ARG2
ADD AREG, &ARG3
MEND
START
LOOP INCR A,B,C
LABEL INCR DATA, DATA2,DATA3
A DC 2 B DC 2 C DS 2 DATA1 DC 3 DATA2 DS 2 DATA3 DC 4 END

MDTC →

(MDT) (Macro Definition Table)	
Index	Card
01	&LAB INCR &ARG1, &ARG2,&ARG3

MNTC →

(MNT)Macro Name Table		
Index	Macro Name	MDT Index
01	INCR	01

(ALA) (Argument List Array)	
Index	Argument
01	&LAB
02	&ARG1
03	&ARG2
04	&ARG3

Pass2 of 2 Pass Macro Processor

Intermediate code	Expanded code
START	START
LOOP INCR A,B,C	
LABEL INCR DATA, DATA2,DATA3	
A DC 2 B DC 2 C DS 2 DATA1 DC 3 DATA2 DS 2 DATA3 DC 4 END	

(MDT) (Macro Definition Table)	
Index	Card
01	&LAB INCR &ARG1, &ARG2,&ARG3
02	#1 ADD AREG, #2
03	ADD AREG, #3
04	ADD AREG, #4
05	MEND

(MNT)Macro Name Table		
Index	Macro Name	MDT Index
01	INCR	01

(ALA) (Argument List Array)	
Index	Argument
01	LOOP
02	A
03	B
04	C

Output:

Output of Pass1 of 2 Pass Macro Processor

Intermediate code

START
LOOP INCR A,B,C
LABEL INCR DATA, DATA2,DATA3
A DC 2 B DC 2 C DS 2 DATA1 DC 3 DATA2 DS 2 DATA3 DC 4 END

MDTC

(MDT) (Macro Definition Table)	
Index	Card
01	&LAB INCR &ARG1, &ARG2,&ARG3
02	#1 ADD AREG, #2
03	ADD AREG, #3
04	ADD AREG, #4
05	MEND

MNTC

(MNT)Macro Name Table		
Index	Macro Name	MDT Index
01	INCR	01

(ALA) (Argument List Array)	
Index	Argument
01	&LAB
02	&ARG1
03	&ARG2
04	&ARG3

Output of Pass2 of 2 Pass Macro Processor

MDTP

(MDT) (Macro Definition Table)	
Index	Card
01	&LAB INCR &ARG1, &ARG2,&ARG3
02	#1 ADD AREG, #2
03	ADD AREG, #3
04	ADD AREG, #4
05	MEND

Intermediate code	Expanded code
START	START

LOOP INCR A,B,C	LOOP ADD AREG, A
	ADD AREG, B
	ADD AREG, C

LABEL INCR DATA, DATA2,DATA3	LABEL ADD AREG, DATA
	ADD AREG, DATA2
	ADD AREG, DATA3

A DC 2 B DC 2 C DS 2 DATA1 DC 3 DATA2 DS 2 DATA3 DC 4 END	A DC 2 B DC 2 C DS 2 DATA1 DC 3 DATA2 DS 2 DATA3 DC 4 END

(MNT)Macro Name Table		
Index	Macro Name	MDT Index
01	INCR	01

(ALA) (Argument List Array)	
Index	Argument
01	LABEL
02	DATA
03	DATA2
04	DATA3

Software Requirement:

1. Fedora
2. Eclipse
3. JDK

Hardware Requirement:
Not Specific

Conclusion: We have successfully completed implementation of Pass-I and Pass-II of two pass macro processor.

Assignment No.: 03

Problem Statement:
Write a program to create a Dynamic Link Library for any mathematical operations (arithmetic, trigonometric and string operation) and write an application program to test it. (Java Native Interface/Use VB/VC++)

Objectives:
1. To study and understand concept of DLL.
2. To understand VC++.
3. To implement DLL using VC++.

Theory:

Dynamic linking is a mechanism that links applications to libraries at run time. The libraries remain in their own files and are not copied into the executable files of the applications. DLLs link to an application when the application is run, rather than when it is created. DLLs may contain links to other DLLs.

Many times, DLLs are placed in files with different extensions such as **.exe**, **.drv** or **.dll**.

Advantages of DLL

Given below are a few advantages of having DLL files.

Uses fewer resources

DLL files don't get loaded into the RAM together with the main program; they don't occupy space unless required. When a DLL file is needed, it is loaded and run. For example, as long as a user of Microsoft Word is editing a document, the printer DLL file is not required in RAM. If the user decides to print the document, then the Word application causes the printer DLL file to be loaded and run.

Promotes modular architecture

A DLL helps promote developing modular programs. It helps you develop large programs that require multiple language versions or a program that requires modular architecture. An example of a modular program is an accounting program having many modules that can be dynamically loaded at run-time.

Aid easy deployment and installation

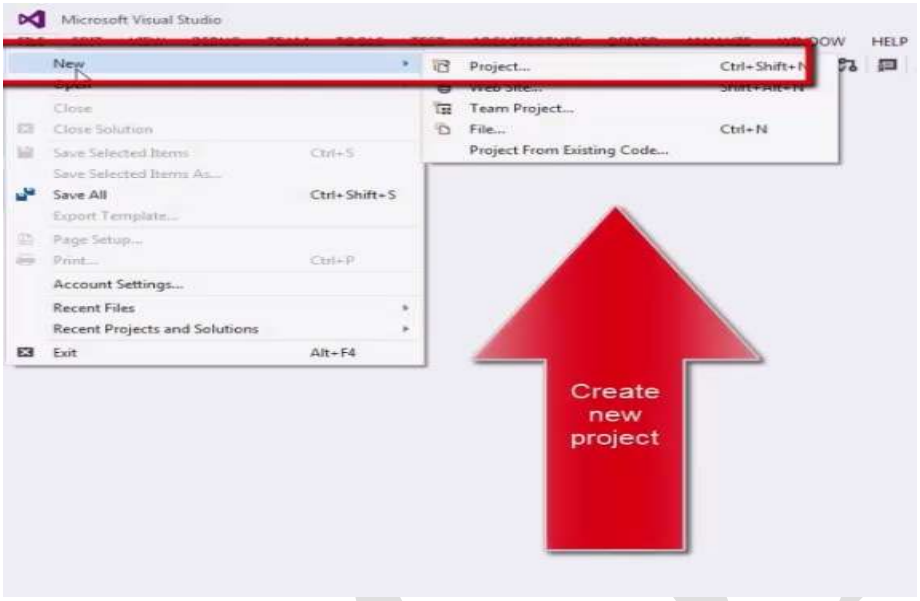
When a function within a DLL needs an update or a fix, the deployment and installation of the DLL does not require the program to be relinked with the DLL. Additionally, if multiple programs use the same DLL, then all of them get benefited from the update or the fix. This issue may occur more frequently when you use a third-party DLL that is regularly updated or fixed.

Applications and DLLs can link to other DLLs automatically, if the DLL linkage is specified in the IMPORTS section of the module definition file as a part of the compile. Else, you can explicitly load them using the Windows LoadLibrary function.

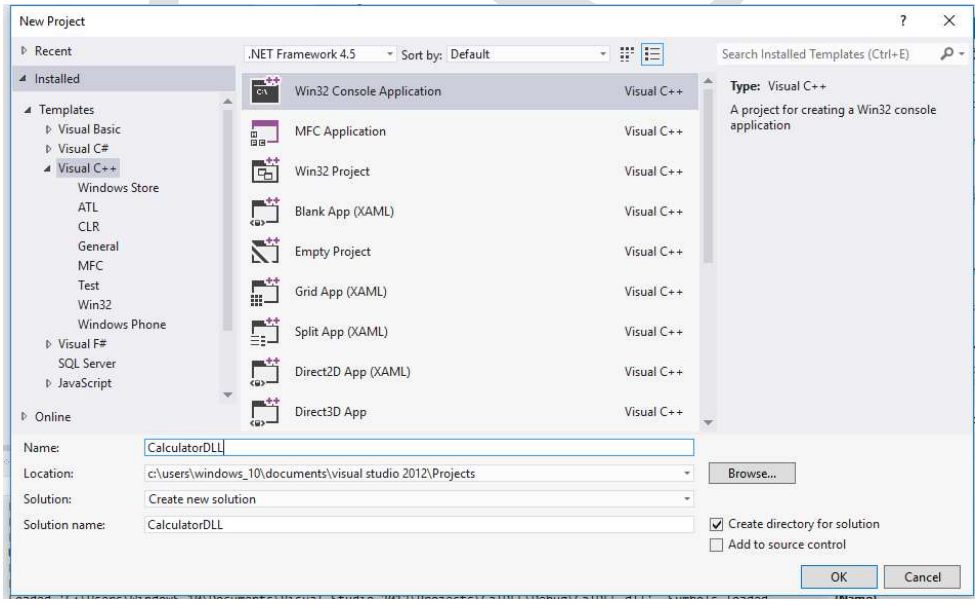
Algorithm/Flow Chart:

Steps to create DLL in C++

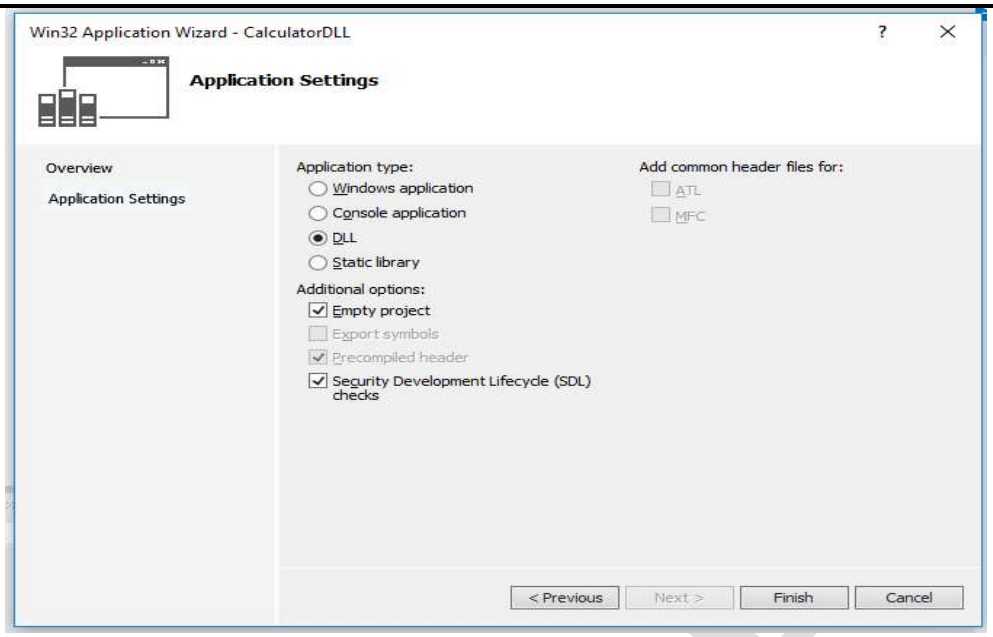
- Open the visual studio and click on the menu bar to create a new project. See the below Image.



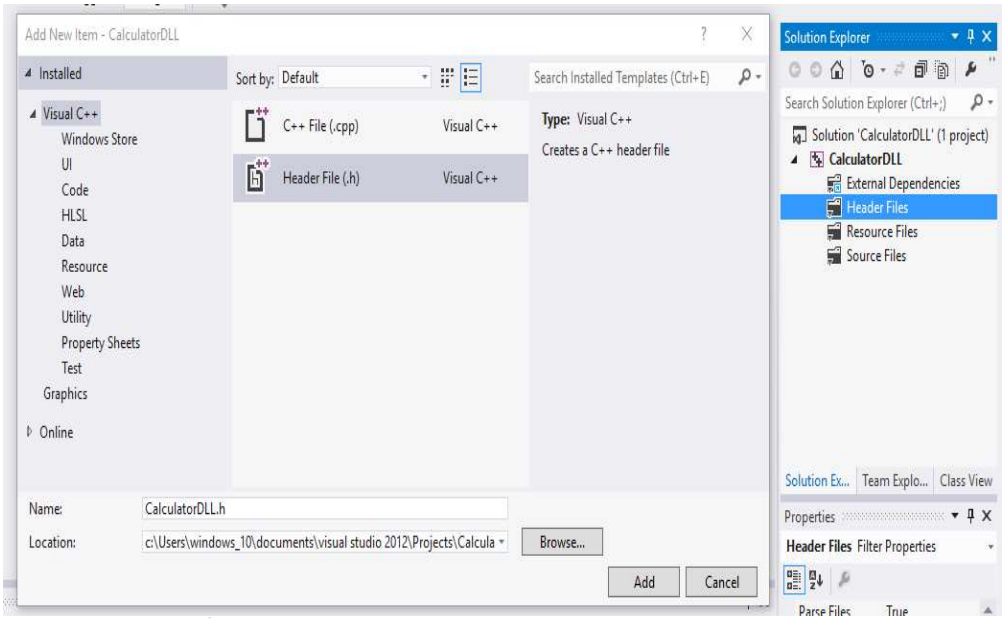
- After selecting the new project, a new dialog box will be open, here select the project type Win32 and give the name to the DLL project.



- On the Overview page of the Win32 Application Wizard dialog box, choose the Next button. After clicking the next button a new window will open. It is Application setting window here we will select the type of the application and click on the finish button to create the DLL project.



- After creating the DLL project you have to add the header files and source file as per your requirements. Here I am adding only one header file.

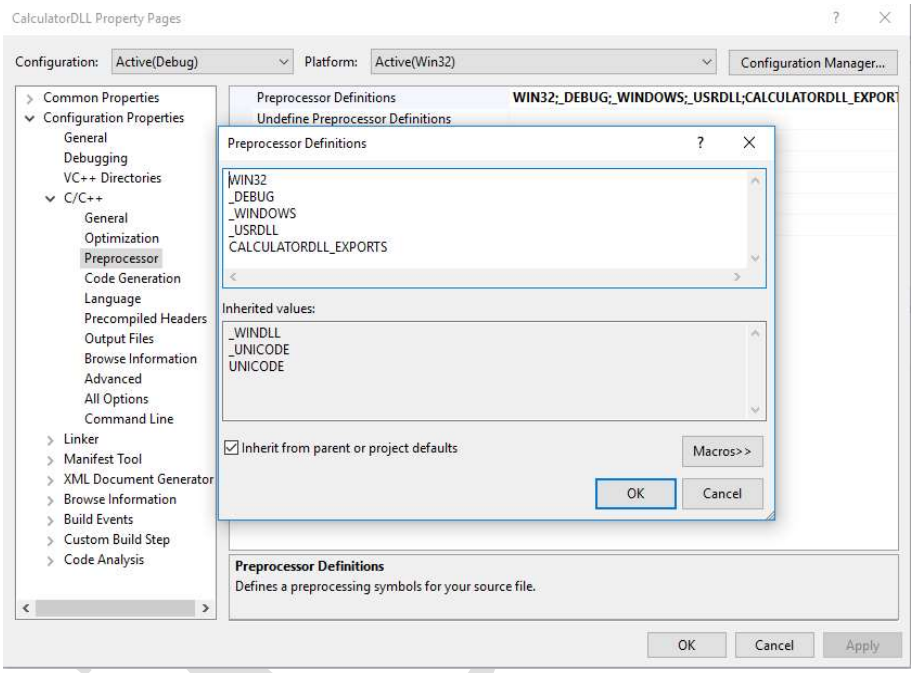


- When you have created the header file then write the desired content as per the requirements. Here I am creating a library that performs some basic arithmetic operation like addition.

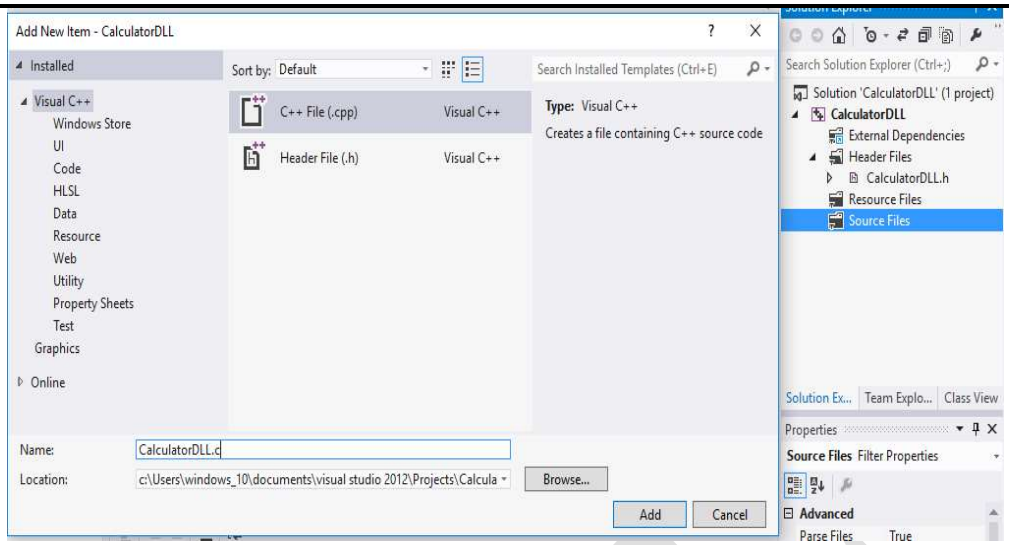
```
#ifndef _CALCULATORDLL_h_
#define _CALCULATORDLL_h_
#ifdef CALCULATORDLL_EXPORTS
#define CALCULATORDLL_API __declspec(dllexport)
```

```
#else
#define CALCULATORDLL_API __declspec(dllimport)
#endif
CALCULATORDLL_API int Addition(int x,int y);
#endif
```

Note: When you have created a DLL project then automatically PROJECTNAME_EXPORTS is defined in preprocessor symbols of the DLL project. In this example, CALCULATIONDLL_EXPORTS is defined when your CALCULATIONDLL DLL project is built.



- Now it's time to define your class member function in the source file. Here I am defining all member functions in CalculatorDLL.C file.



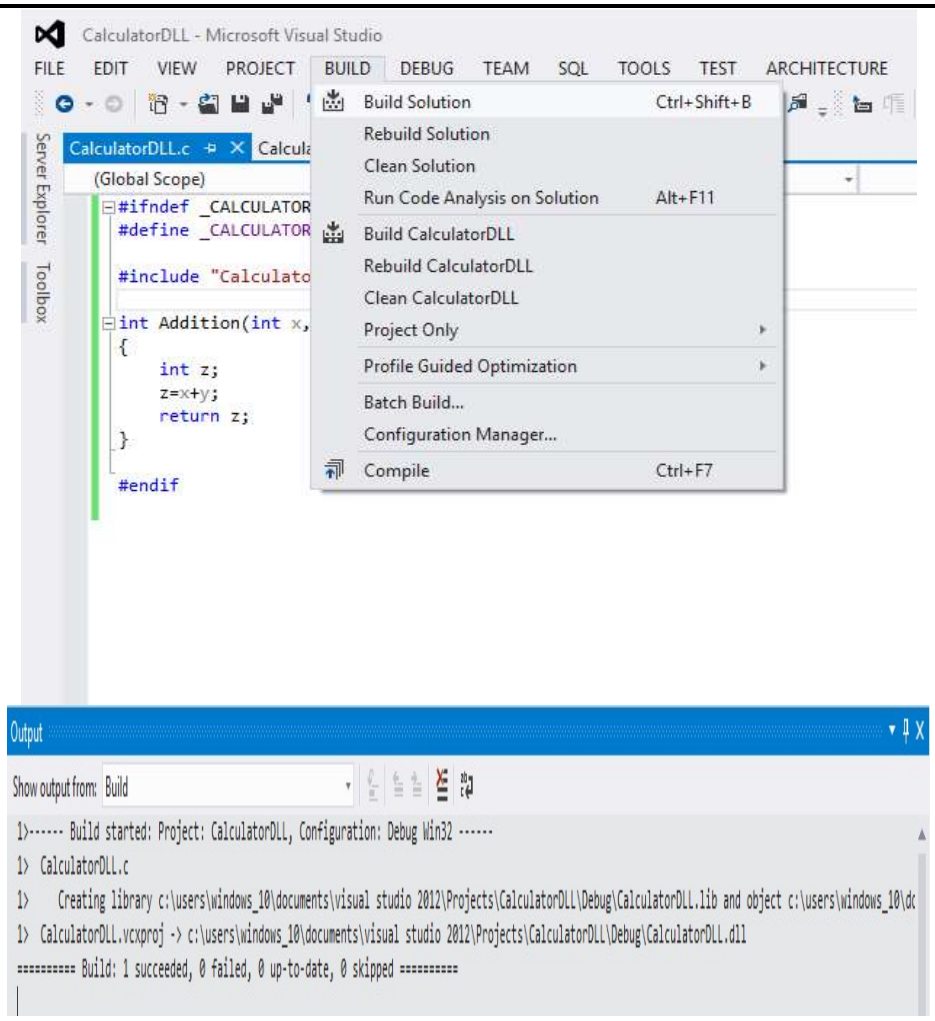

```
#ifndef _CALCULATORDLL_c_
#define _CALCULATORDLL_c_

#include "CalculatorDLL.h"

int Addition(int x,int y)
{
	int z;
	z=x+y;
	return z;
}


#endif
```

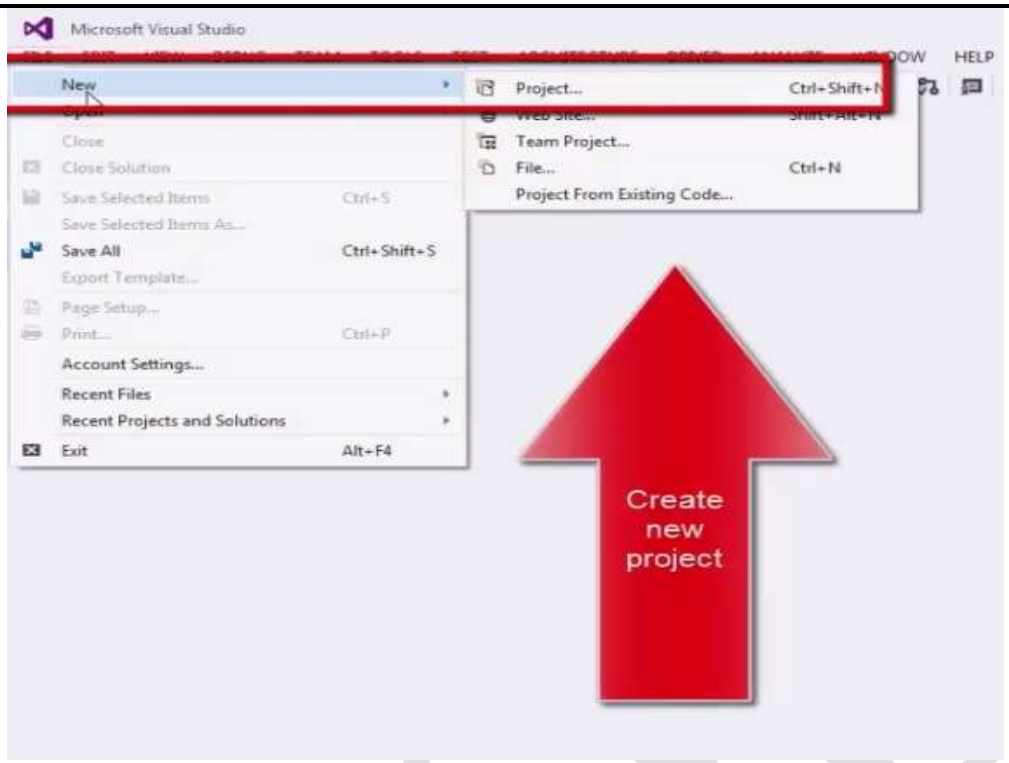
Now source and header files are added to the DLL project, to create the DLL and lib just build the DLL project. If everything is fine and your DLL project compiles perfectly without any error then a DLL and .lib file will be generated.



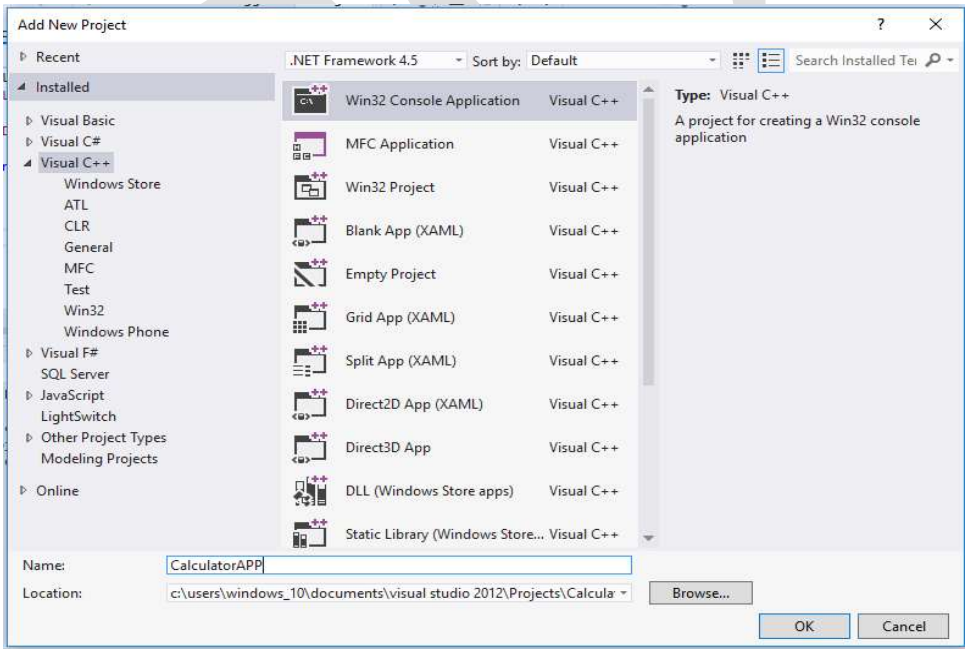
Steps to create a C ++ Application

Here I am creating a c++ application that will use the created DLL.

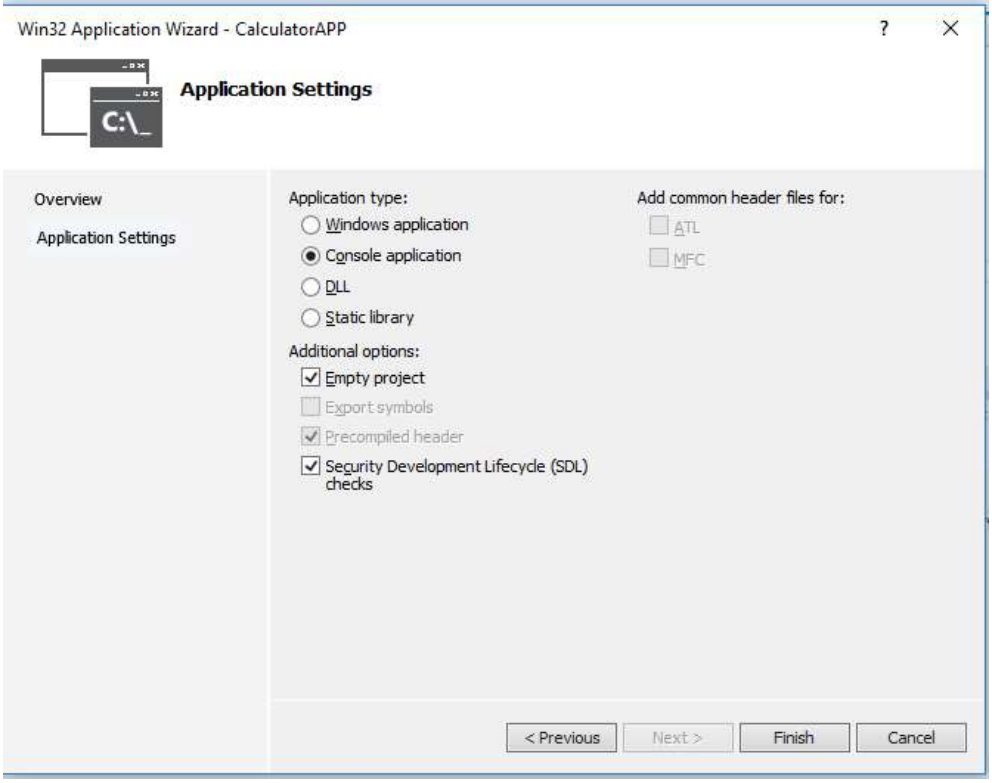
- Click on the menu bar to create a new c++ Application project that uses the DLL which I have created just now.



- After selecting the new project a new dialog box will be open, here select the project type Win32 Console Application and give the name to the App project.



- On the Overview page of the Win32 Application Wizard dialog box, choose the Next button. After clicking the next button a new window will open. It is the Application setting window here we will select the type of the application and click on the finish button to create the c++ Console Application project.

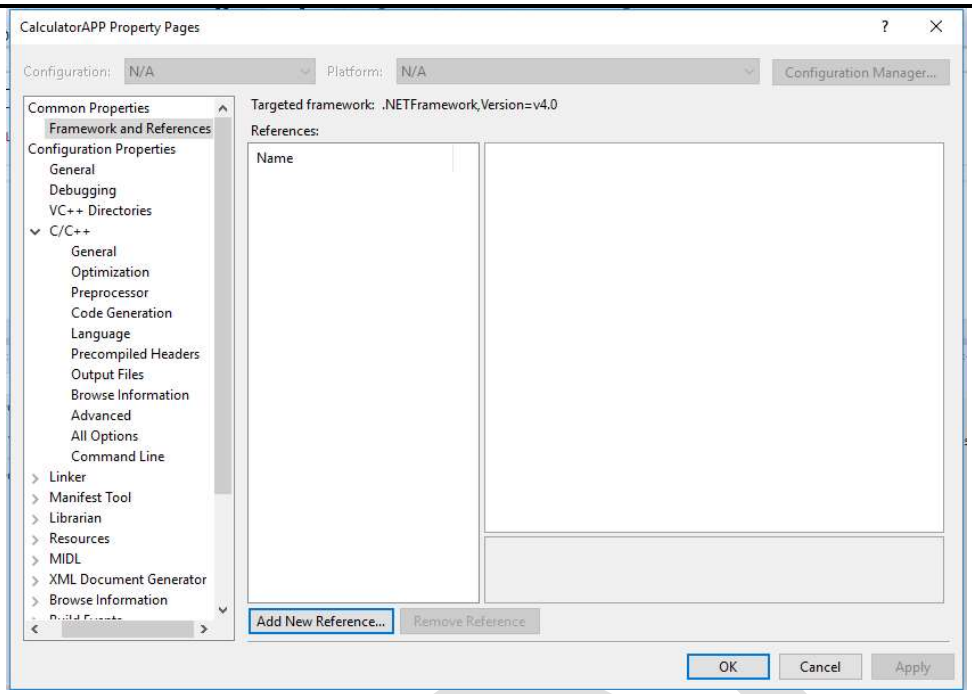


Now your C++ application project is ready to use the DLL (Dynamic linking library).

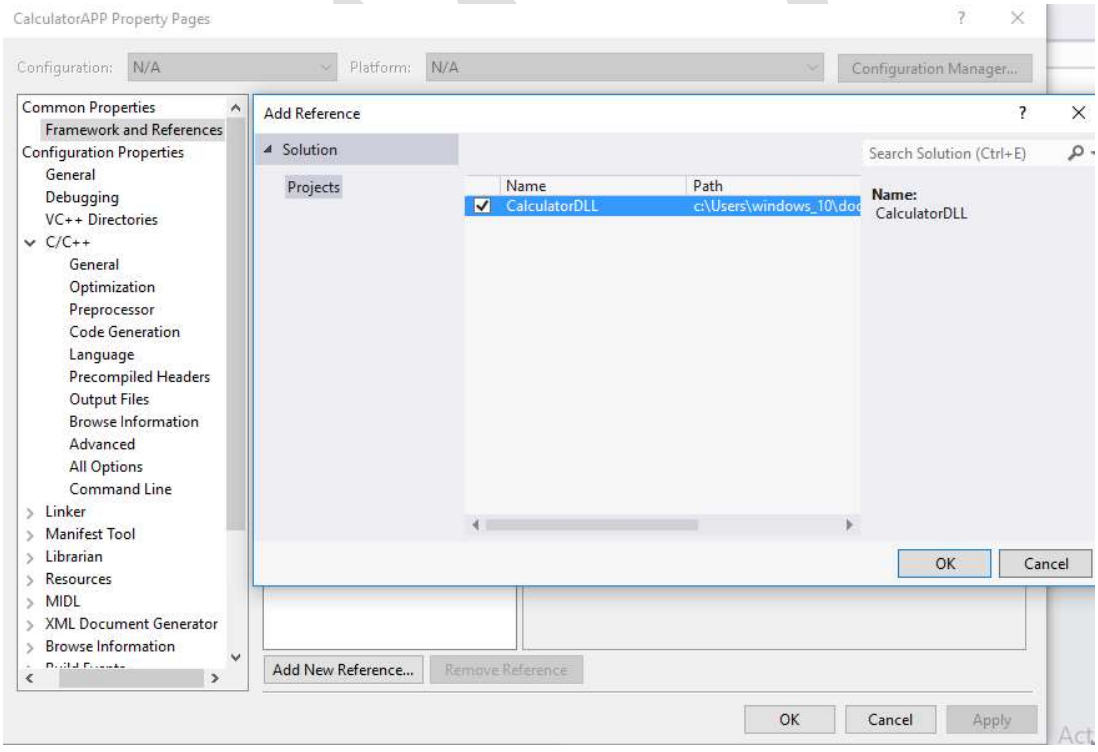
How to Link DLL with c++ Application

Here I am discussing simple steps to link the DLL project with the C++ Application project.

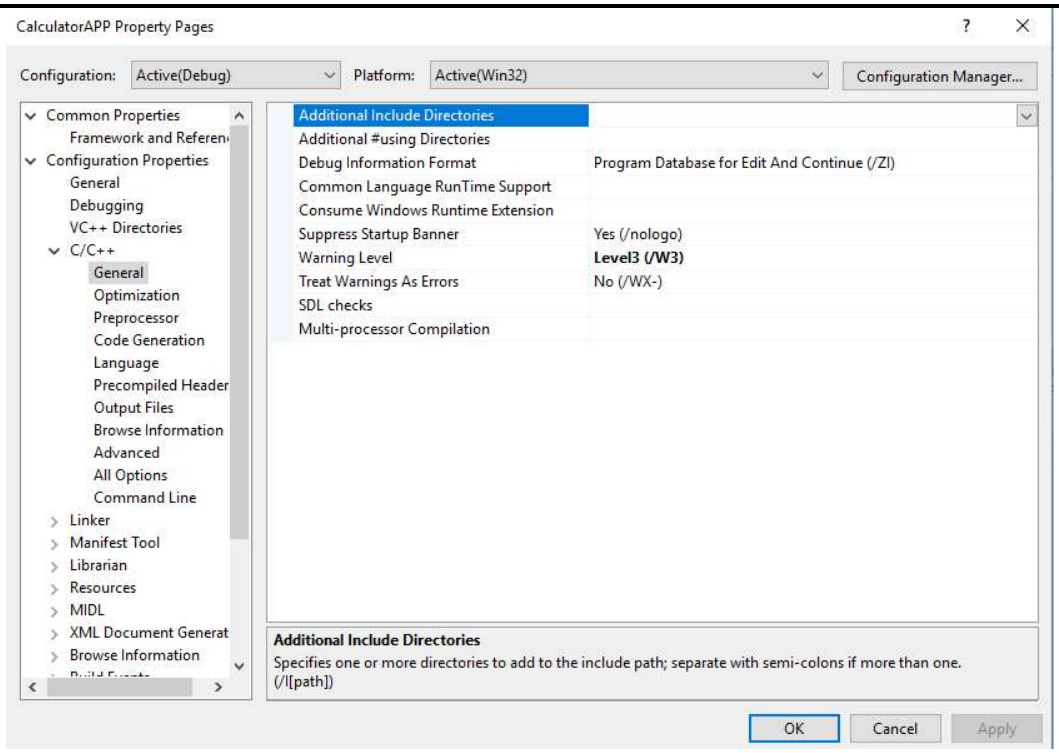
- When we have created the DLL and Application then after that we have to reference the DLL to the Application that makes the enable to Application to use the DLL function as per the requirement. To do this, under the CalculatorAPP project in Solution Explorer, select the References item. On the menu bar, choose Project, Add Reference.

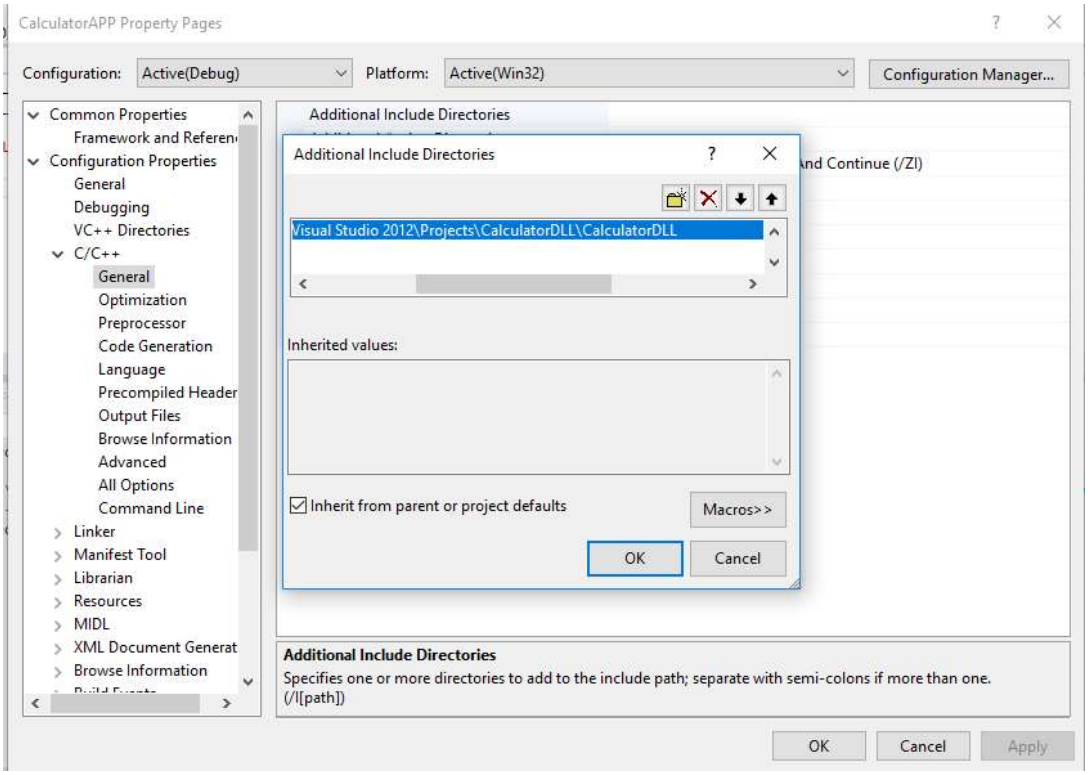


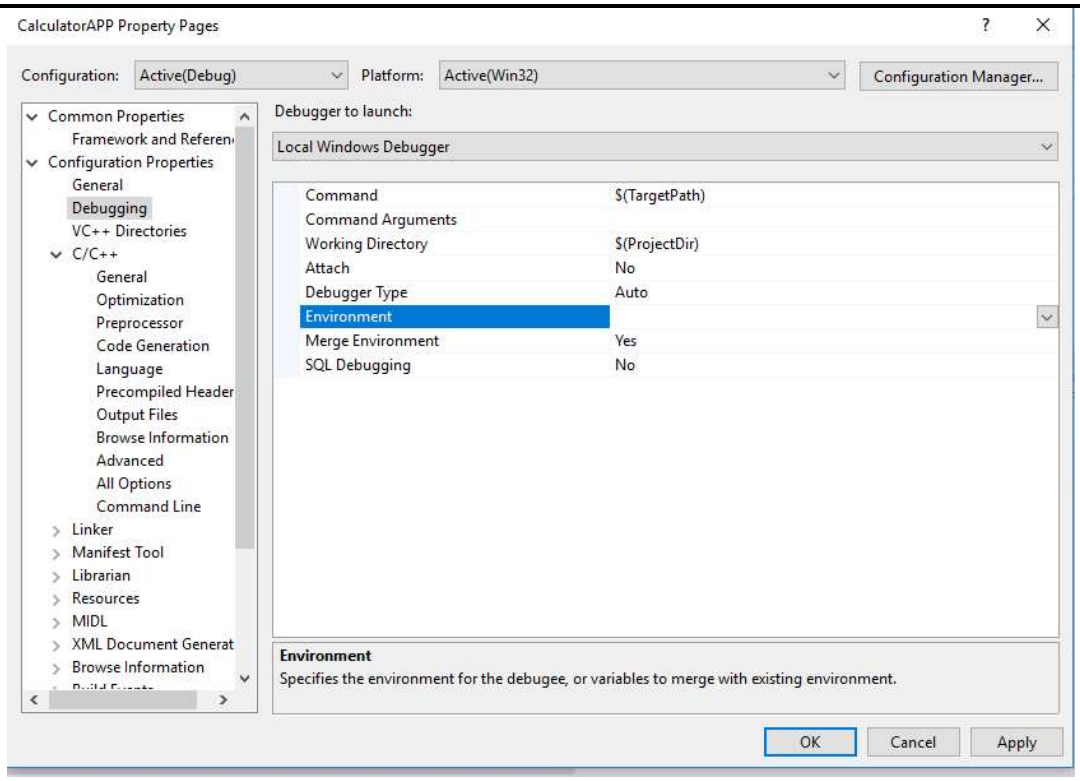
- When you click on the Add new Reference then a dialog box will be open which has the lists of the library that you can reference. You need to just click on the check button to the required library. Here only one library is showing in the dialog box.

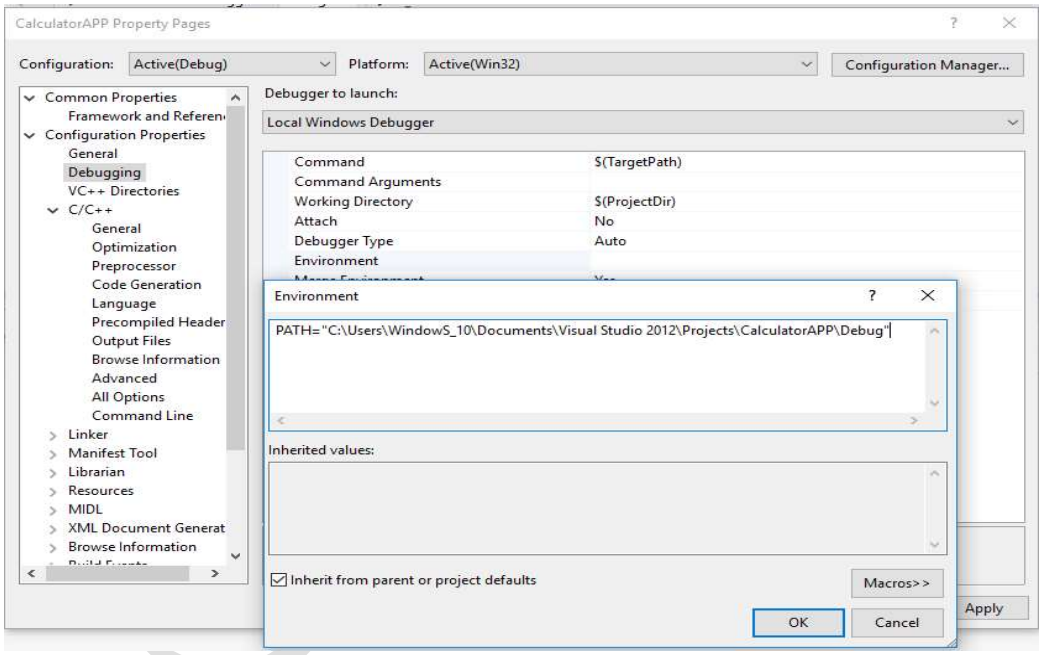


- Now your created library is linked with the created Application, but before using the DLL in Application you have to add the DLL header file. We just reference the DLL header file to give the path of original DLL header files in Application project included directories path.

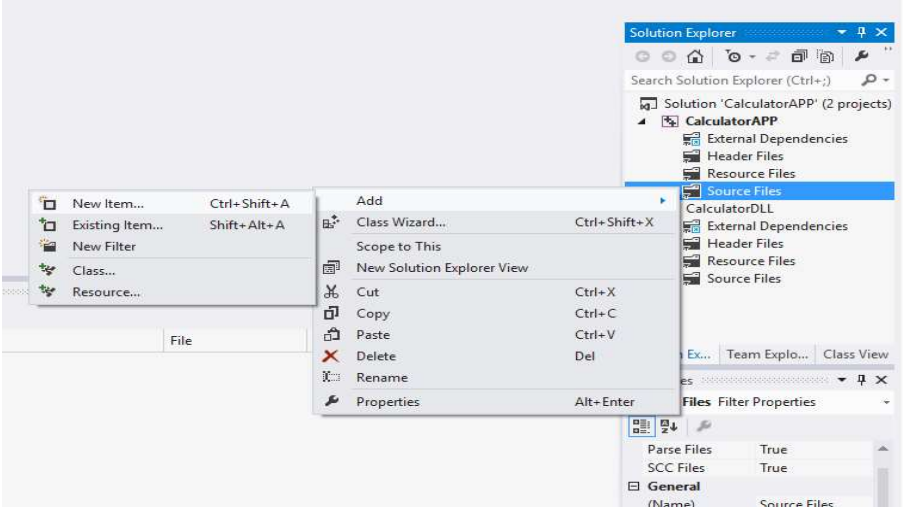

CalculatorAPP Property Pages
Configuration: Active(Debug)
Platform: Active(Win32)
Configuration Manager...
Common Properties
Framework and Referen
Configuration Properties
General
Debugging
VC++ Directories
C/C++
General
Optimization
Preprocessor
Code Generation
Language
Precompiled Header
Output Files
Browse Information
Advanced
All Options
Command Line
Linker
Manifest Tool
Librarian
Resources
MIDL
XML Document Generat
Browse Information
Additional Include Directories
Additional #using Directories
Debug Information Format
Program Database for Edit And Continue (/ZI)
Common Language RunTime Support
Consume Windows Runtime Extension
Suppress Startup Banner
Yes (/nologo)
Warning Level
Level3 (/W3)
Treat Warnings As Errors
No (/WX-)
SDL checks
Multi-processor Compilation
Additional Include Directories
Specifies one or more directories to add to the include path; separate with semi-colons if more than one.
(/I[path])
OK
Cancel
Apply

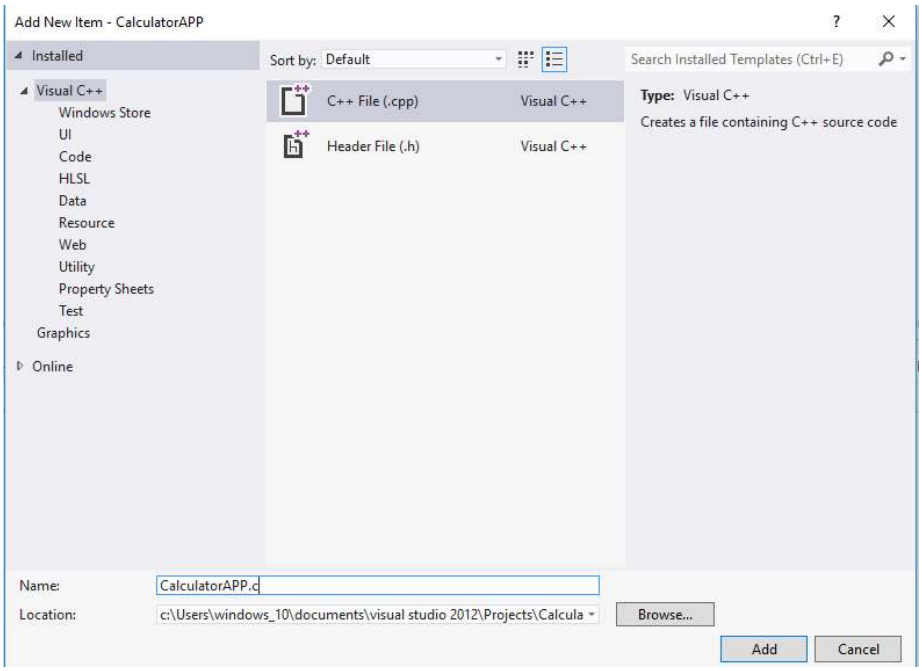

CalculatorAPP Property Pages
Configuration: Active(Debug)
Platform: Active(Win32)
Configuration Manager...
Common Properties
Framework and Referen
Configuration Properties
General
Debugging
VC++ Directories
C/C++
General
Optimization
Preprocessor
Code Generation
Language
Precompiled Header
Output Files
Browse Information
Advanced
All Options
Command Line
Linker
Manifest Tool
Librarian
Resources
MIDL
XML Document Generat
Browse Information
Additional Include Directories
Additional Include Directories
Visual Studio 2012\Projects\CalculatorDLL\CalculatorDLL
Inherited values:
Inherit from parent or project defaults
Macros>>
OK
Cancel
Additional Include Directories
Specifies one or more directories to add to the include path; separate with semi-colons if more than one.
(/I[path])
OK
Cancel
Apply





- Now it's time to define your class member function in the source file. Here I am calling all member functions in CalculatorAPP.C file.

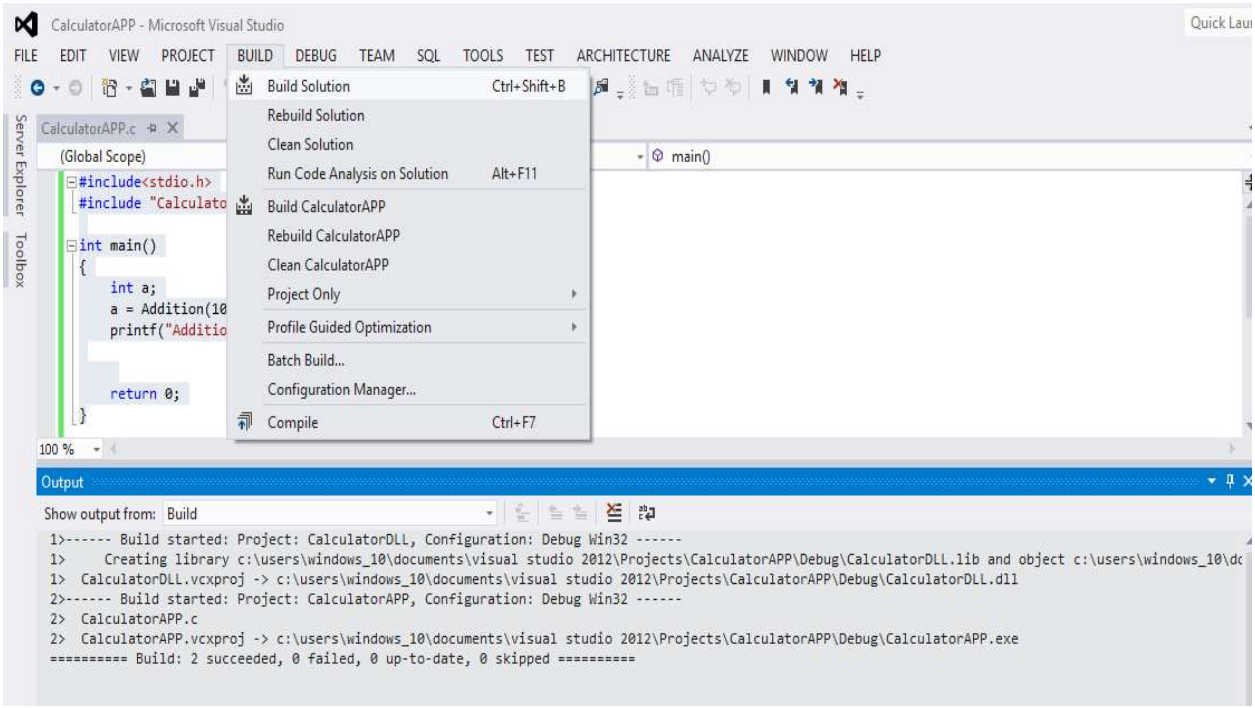


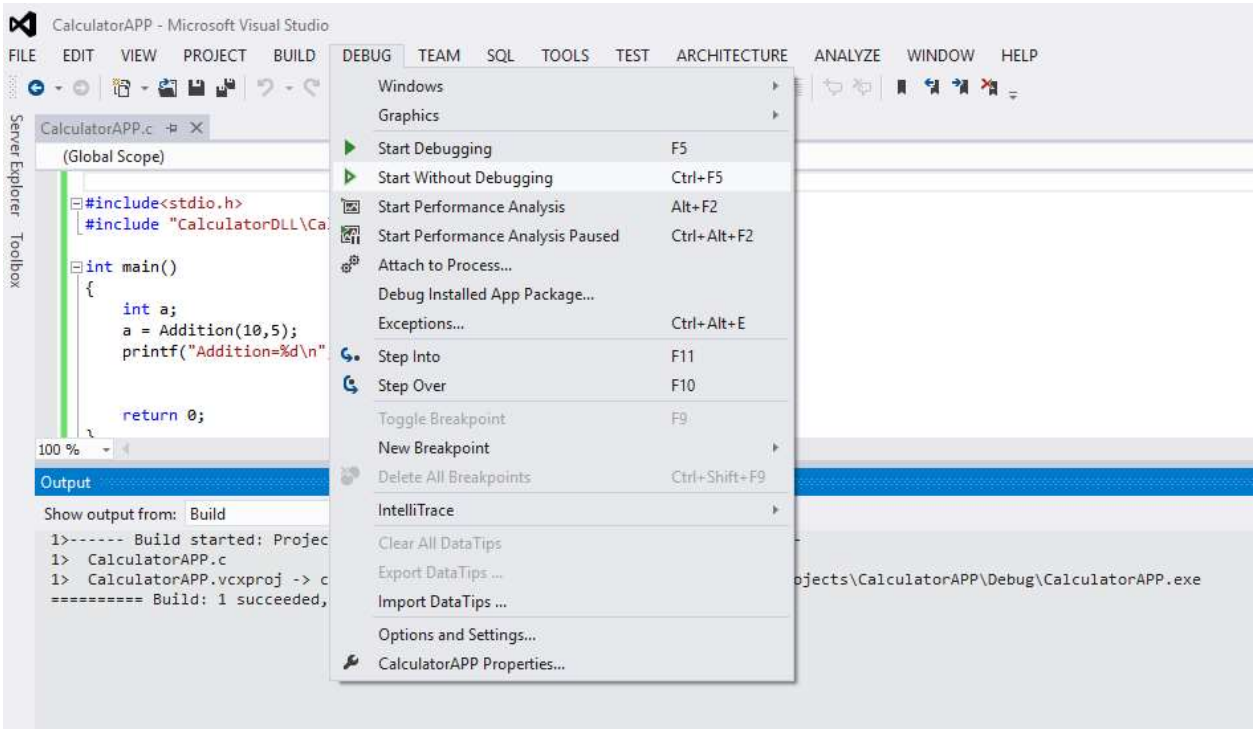


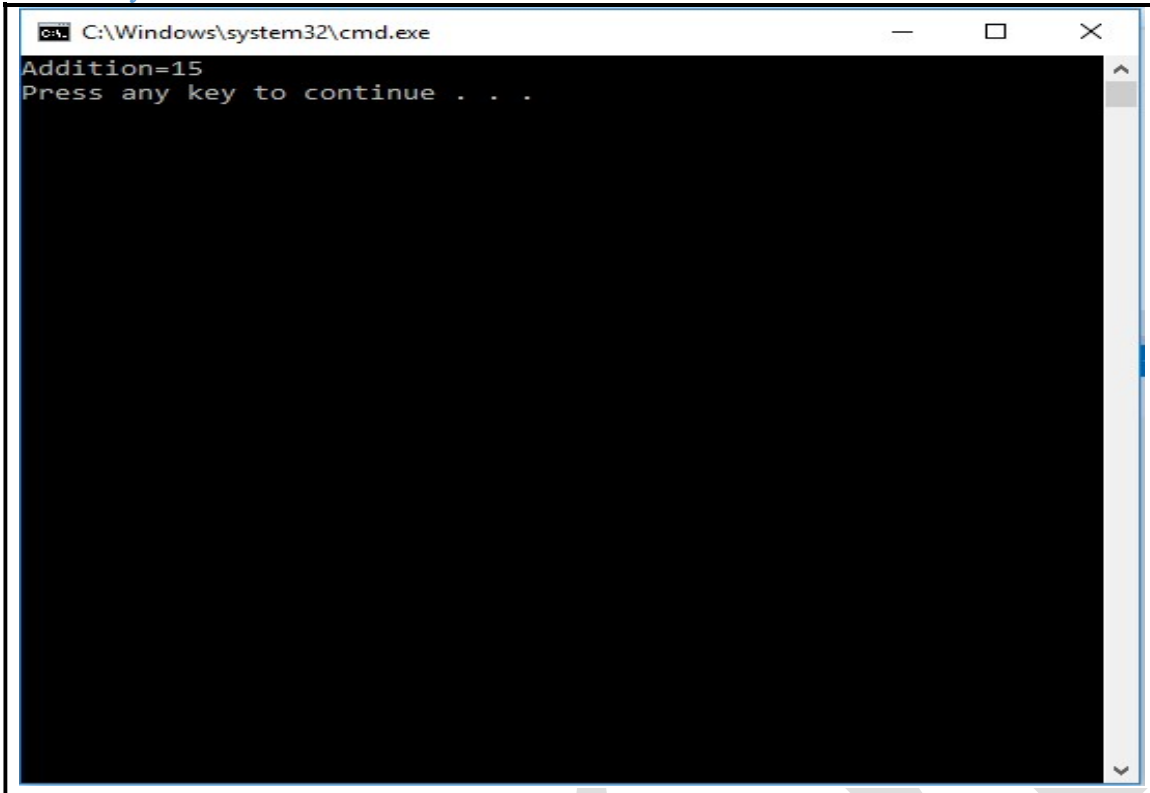

```
#include<stdio.h>
#include "CalculatorDLL\CalculatorDLL.h"

int main()
{
        int a;
        a = Addition(10,5);
        printf("Addition=%d\n",a);
```

```
        return 0;
}
```


CalculatorAPP - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Build Solution Ctrl+Shift+B
Rebuild Solution
Clean Solution
Run Code Analysis on Solution Alt+F11
Build CalculatorAPP
Rebuild CalculatorAPP
Clean CalculatorAPP
Project Only
Profile Guided Optimization
Batch Build...
Configuration Manager...
Compile Ctrl+F7
CalculatorAPP.c
(Global Scope)
main()
#include<stdio.h>
#include "Calculato
int main()
{
int a;
a = Addition(10
printf("Additio
return 0;
}
100 %
Output
Show output from: Build
1>------ Build started: Project: CalculatorDLL, Configuration: Debug Win32 ------
1> Creating library c:\users\windows_10\documents\visual studio 2012\Projects\CalculatorAPP\Debug\CalculatorDLL.lib and object c:\users\windows_10\dc
1> CalculatorDLL.vcxproj -> c:\users\windows_10\documents\visual studio 2012\Projects\CalculatorAPP\Debug\CalculatorDLL.dll
2>------ Build started: Project: CalculatorAPP, Configuration: Debug Win32 ------
2> CalculatorAPP.c
2> CalculatorAPP.vcxproj -> c:\users\windows_10\documents\visual studio 2012\Projects\CalculatorAPP\Debug\CalculatorAPP.exe
========== Build: 2 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========


CalculatorAPP - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Windows
Graphics
Start Debugging F5
Start Without Debugging Ctrl+F5
Start Performance Analysis Alt+F2
Start Performance Analysis Paused Ctrl+Alt+F2
Attach to Process...
Debug Installed App Package...
Exceptions... Ctrl+Alt+E
Step Into F11
Step Over F10
Toggle Breakpoint F9
New Breakpoint
Delete All Breakpoints Ctrl+Shift+F9
IntelliTrace
Clear All DataTips
Export DataTips ...
Import DataTips ...
Options and Settings...
CalculatorAPP Properties...
CalculatorAPP.c
(Global Scope)
#include<stdio.h>
#include "CalculatorDLL\Ca
int main()
{
int a;
a = Addition(10,5);
printf("Addition=%d\n"
return 0;
100 %
Output
Show output from: Build
1>------ Build started: Projec
1> CalculatorAPP.c
1> CalculatorAPP.vcxproj -> c
========== Build: 1 succeeded,
ojects\CalculatorAPP\Debug\CalculatorAPP.exe



Input:
Enter Number1:10
Enter Number2:05

Output:

Choose Operation:
1.Addition.
2.Substraction.
3.Multiplication.
4.Division.
Enter Your Choice:1
Addition:15

Choose Operation:
1.Addition.
2.Substraction.
3.Multiplication.
4.Division.
Enter Your Choice:2
Substration:05

Choose Operation:
1.Addition.
2.Substraction.
3.Multiplication.
4.Division.
Enter Your Choice:3

Mutiplication:150

Choose Operation:
1.Addition.
2.Substraction.
3.Multiplication.
4.Division.
Enter Your Choice:4
Division:02

Software Requirement:

1. Windows.
2. Visual Studio.

Hardware Requirement:
Not Specific

Conclusion:
Successfully implemented DLL and tested it with VC++

Assignment No.: 4

Problem Statement:
Write a program to solve classical problems of synchronization using mutex and semaphore.

Objectives:
1. To understand reader writer synchronization problem
2. To solve reader-writer synchronization problem using mutex and semaphore

Theory:
Process Synchronization was introduced to handle problems that arise while multiple process executions.

Process is categorized into two types on the basis of synchronization and these are given below:

- Independent Process
- Cooperative Process

Independent Processes

Two processes are said to be independent if the execution of one process does not affect the execution of another process.

Cooperative Processes

Two processes are said to be cooperative if the execution of one process affects the execution of another process. These processes need to be synchronized so that the order of execution can be guaranteed.

Process Synchronization

It is the task phenomenon of coordinating the execution of processes in such a way that no two processes can have access to the same shared data and resources.

- It is a procedure that is involved in order to preserve the appropriate order of execution of cooperative processes.
- In order to synchronize the processes, there are various synchronization mechanisms.
- Process Synchronization is mainly needed in a multi-process system when multiple processes are running together, and more than one processes try to gain access to the same shared resource or any data at the same time.

Race Condition

At the time when more than one process is either executing the same code or accessing the same memory or any shared variable; In that condition, there is a possibility that the output or the value of the shared variable is wrong so for that purpose all the processes are doing the race to say that my output is correct. This condition is commonly known as **a race condition.** As several processes access

and process the manipulations on the same data in a concurrent manner and due to which the outcome depends on the particular order in which the access of data takes place.

Mainly this condition is a situation that may occur inside the **critical section**. Race condition in the critical section happens when the result of multiple thread execution differs according to the order in which the threads execute. But this condition is critical sections can be avoided if the critical section is treated as an atomic instruction. Proper thread synchronization using locks or atomic variables can also prevent race conditions.

Critical Section Problem

A Critical Section is a code segment that accesses shared variables and has to be executed as an atomic action. It means that in a group of cooperating processes, at a given point of time, only one process must be executing its critical section. If any other process also wants to execute its critical section, it must wait until the first one finishes. The entry to the critical section is mainly handled by wait() function while the exit from the critical section is controlled by the signal() function.

Semaphore can be used in other synchronization problems besides Mutual Exclusion.

Below are some of the classical problem depicting flaws of process synchronaization in systems where cooperating processes are present.

We will discuss the following three problems:

1. Bounded Buffer (Producer-Consumer) Problem
2. Dining Philosophers Problem
3. The Readers Writers Problem

Bounded Buffer Problem

Because the buffer pool has a maximum size, this problem is often called the **Bounded buffer problem**.

- This problem is generalised in terms of the **Producer Consumer problem**, where a **finite** buffer pool is used to exchange messages between producer and consumer processes.
- Solution to this problem is, creating two counting semaphores "full" and "empty" to keep track of the current number of full and empty buffers respectively.
- In this Producers mainly produces a product and consumers consume the product, but both can use of one of the containers each time.
- The main complexity of this problem is that we must have to maintain the count for both empty and full containers that are available.

Dining Philosophers Problem

- The dining philosopher's problem involves the allocation of limited resources to a group of processes in a deadlock-free and starvation-free manner.
- There are five philosophers sitting around a table, in which there are five chopsticks/forks kept beside them and a bowl of rice in the centre, When a philosopher wants to eat, he uses two

chopsticks - one from their left and one from their right. When a philosopher wants to think, he keeps down both chopsticks at their original place.

The Readers Writers Problem

- In this problem there are some processes(called **readers**) that only read the shared data, and never change it, and there are other processes(called **writers**) who may change the data in addition to reading, or instead of reading it.
- There are various type of readers-writers problem, most centred on relative priorities of readers and writers.
- The main complexity with this problem occurs from allowing more than one reader to access the data at the same time.

Algorithm/Flowchart:

Algorithm for Reader Writer:

1. import java.util.concurrent.Semaphore;

2. Create a class RW

3. Declare semaphores – mutex and wrt

4. Declare integer variable readcount = 0

5. Create a nested class Reader implements Runnable

a. Override run method (Reader Logic)

i. wait(mutex);
ii. readcount := readcount +1;
iii. if readcount = 1 then
iv. wait(wrt);
v. signal(mutex);
vi. …
vii. reading is performed
viii. …
ix. wait(mutex);
x. readcount := readcount – 1;
xi. if readcount = 0 then signal(wrt);
xii. signal(mutex):

6. Create a nested class Writer implements Runnable

a. Override run method (Writer Logic)

i. wait(wrt);
ii. …
iii. writing is performed
iv. …
v. signal(wrt);

7. Create a class main

a. Create Threads for Reader and Writer

Start these thread

Input:
1. Number of Readers
2. Number of Writers

Output:
Execution of Readers and Writers

Software Requirement:
1. Fedora
2. Eclipse
3. JDK

Hardware Requirement:
Not Specific

Conclusion: Implemented Reader Writer synchronization problem using semaphores in Java

Assignment No.: 05

Problem Statement:
Write a program to simulate CPU Scheduling Algorithms: FCFS, SJF (Preemptive), Priority(Non Preemptive) and Round Robin (Preemptive).

Objectives:
1. To study the process management and various scheduling policies viz. Preemptive and Non preemptive.
2. To study and analyze different scheduling algorithms.

Theory :

The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.

Process scheduling is an essential part of a Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.

Categories of Scheduling

There are two categories of scheduling:

1. **Non-preemptive:** Here the resource can't be taken from a process until the process completes execution. The switching of resources occurs when the running process terminates and moves to a waiting state.
2. **Preemptive:** Here the OS allocates the resources to a process for a fixed amount of time. During resource allocation, the process switches from running state to ready state or from waiting state to ready state. This switching occurs as the CPU may give priority to other processes and replace the process with higher priority with the running process.

A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. There are six popular process scheduling algorithms which we are going to discuss in this chapter −

- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-Next (SJN) Scheduling
- Priority Scheduling
- Shortest Remaining Time
- Round Robin(RR) Scheduling
- Multiple-Level Queues Scheduling

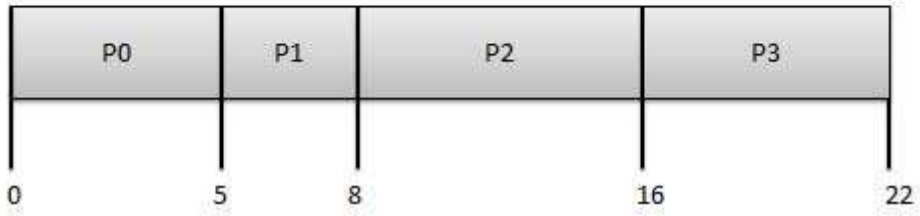
These algorithms are either **non-preemptive or preemptive**. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

First Come First Serve (FCFS)

- Jobs are executed on first come, first serve basis.
- It is a non-preemptive, pre-emptive scheduling algorithm.

- Easy to understand and implement.
- Its implementation is based on FIFO queue.
- Poor in performance as average wait time is high.

Process	Arrival Time	Execute Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	8
P3	3	6	16



Wait time of each process is as follows −

Process	Wait Time : Service Time - Arrival Time
P0	0 - 0 = 0
P1	5 - 1 = 4
P2	8 - 2 = 6
P3	16 - 3 = 13

Average Wait Time: (0+4+6+13) / 4 = 5.75

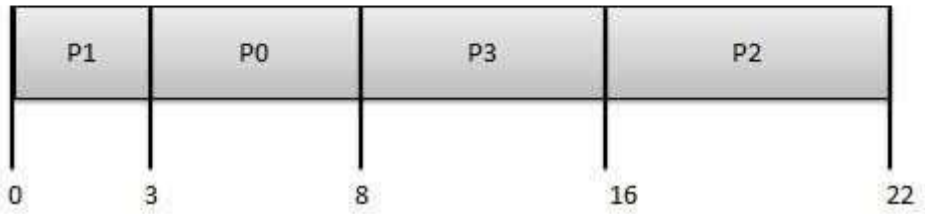
Shortest Job Next (SJN)

- This is also known as **shortest job first**, or SJF
- This is a non-preemptive, pre-emptive scheduling algorithm.
- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.
- Impossible to implement in interactive systems where required CPU time is not known.
- The processer should know in advance how much time process will take.

Given: Table of processes, and their Arrival time, Execution time

Process	Arrival Time	Execution Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	14
P3	3	6	8

Process	Arrival Time	Execute Time	Service Time
P0	0	5	3
P1	1	3	0
P2	2	8	16
P3	3	6	8



Waiting time of each process is as follows −

Process	Waiting Time
P0	0 - 0 = 0
P1	5 - 1 = 4
P2	14 - 2 = 12
P3	8 - 3 = 5

Average Wait Time: (0 + 4 + 12 + 5)/4 = 21 / 4 = 5.25

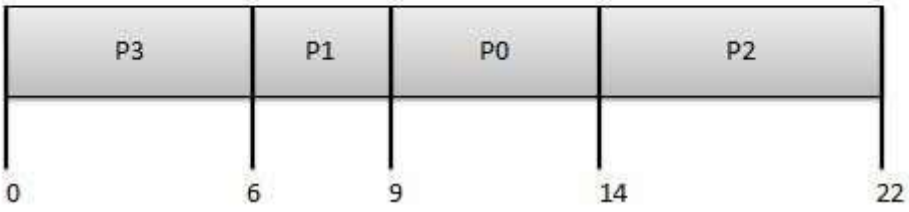
Priority Based Scheduling

- Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
- Each process is assigned a priority. Process with highest priority is to be executed first and so on.
- Processes with same priority are executed on first come first served basis.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Given: Table of processes, and their Arrival time, Execution time, and priority. Here we are considering 1 is the lowest priority.

Process	Arrival Time	Execution Time	Priority	Service Time
P0	0	5	1	0
P1	1	3	2	11
P2	2	8	1	14
P3	3	6	3	5

Process	Arrival Time	Execute Time	Priority	Service Time
P0	0	5	1	9
P1	1	3	2	6
P2	2	8	1	14
P3	3	6	3	0



Waiting time of each process is as follows −

Process	Waiting Time
P0	0 - 0 = 0
P1	11 - 1 = 10
P2	14 - 2 = 12
P3	5 - 3 = 2

Average Wait Time: (0 + 10 + 12 + 2)/4 = 24 / 4 = 6

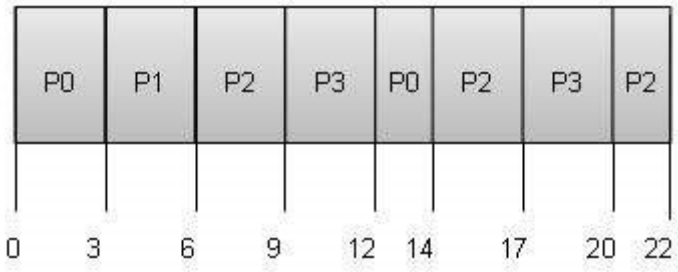
Shortest Remaining Time

- Shortest remaining time (SRT) is the preemptive version of the SJN algorithm.
- The processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion.
- Impossible to implement in interactive systems where required CPU time is not known.
- It is often used in batch environments where short jobs need to give preference.

Round Robin Scheduling

- Round Robin is the preemptive process scheduling algorithm.
- Each process is provided a fix time to execute, it is called a **quantum**.
- Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
- Context switching is used to save states of preempted processes.

Quantum = 3



Wait time of each process is as follows −

Process	Wait Time : Service Time - Arrival Time
P0	(0 - 0) + (12 - 3) = 9
P1	(3 - 1) = 2
P2	(6 - 2) + (14 - 9) + (20 - 17) = 12
P3	(9 - 3) + (17 - 12) = 11

Average Wait Time: (9+2+12+11) / 4 = 8.5

Multiple-Level Queues Scheduling

Multiple-level queues are not an independent scheduling algorithm. They make use of other existing algorithms to group and schedule jobs with common characteristics.

- Multiple queues are maintained for processes with common characteristics.
- Each queue can have its own scheduling algorithms.
- Priorities are assigned to each queue.

For example, CPU-bound jobs can be scheduled in one queue and all I/O-bound jobs in another queue. The Process Scheduler then alternately selects jobs from each queue and assigns them to the CPU based on the algorithm assigned to the queue.

Algorithm/Flowchart:

1. FCFS

a. Input the processes along with their burst time (bt).
b. Find waiting time (wt) for all processes.
c. As first process that comes need not to wait so waiting time for process 1 will be 0 i.e.wt[0] = 0.
d. Find waiting time for all other processes i.e. for process i ->wt[i] = bt[i-1] + wt[i-1] .
e. Find turnaround time = waiting_time + burst_time for all processes.
f. Find average waiting time = total_waiting_time / no_of_processes.
g. Similarly, find average turnaround time = total_turn_around_time / no_of_processes.

2. SJF

a. Traverse until all process gets completely executed.
b. Find process with minimum remaining time at every single time lap.
c. Reduce its time by 1.
d. Check if its remaining time becomes 0
e. Increment the counter of process completion.
f. Completion time of current process = current_time +1;
g. Calculate waiting time for each completed process.wt[i]= Completion time - arrival_time-burst_time
h. Increment time lap by one.
i. Find turnaround time (waiting_time+burst_time).

3. Priority

a. First input the processes with their burst time and priority.
b. Sort the processes, burst time and priority according to the priority.
c. Now simply apply FCFS algorithm.

4. RR

a. Create an array **rem_bt[]** to keep track of remaining burst time of processes. This array is initially a copy of bt[] (burst times array).
b. Create another array **wt[]** to store waiting times of processes.
c. Initialize this array as 0.
d. Initialize time : t = 0
e. Keep traversing the all processes while all processes are not done. Do following for i'thprocess if it is not done yet.

```
If rem_bt[i] > quantum
        t = t + quantum
       bt_rem[i] -= quantum;
Else // Last cycle for this process
        t = t + bt_rem[i];
wt[i] = t - bt[i]
bt_rem[i] = 0; // This process is over
```

Input:

1. Enter the number of processes
2. Enter burst time and arrival time of each process

Output:

1. Compute Waiting time, turnaround time, average waiting time, average turnaround time and throughput.

For each algorithm display result as follows:

Process	Burst Time	Arrival Time	Waiting Time	Turnaround Time
P1				
P2				
P3				

Calculate

1. Average waiting time=
2. Average turnaround time=
3. Throughput=

Software Requirement:

1. Fedora
2. Eclipse
3. JDK

Hardware Requirement:
No Specific

Conclusion:

Thus, CPU scheduling algorithms implemented successfully

Assignment No.: 06

Problem Statement:

Write a program to simulate memory placement strategies

1. First Fit
2. Best Fit
3. Worst Fit
4. Next Fit

Objectives:

1. To acquire knowledge memory placement strategies
2. To be able to implement memory placement strategies

Theory:

What is Main Memory:

The main memory is central to the operation of a modern computer. Main Memory is a large array of words or bytes, ranging in size from hundreds of thousands to billions. Main memory is a repository of rapidly available information shared by the CPU and I/O devices. Main memory is the place where programs and information are kept when the processor is effectively utilizing them. Main memory is associated with the processor, so moving instructions and information into and out of the processor is extremely fast. Main memory is also known as RAM(Random Access Memory). This memory is a volatile memory.RAM lost its data when a power interruption occurs.

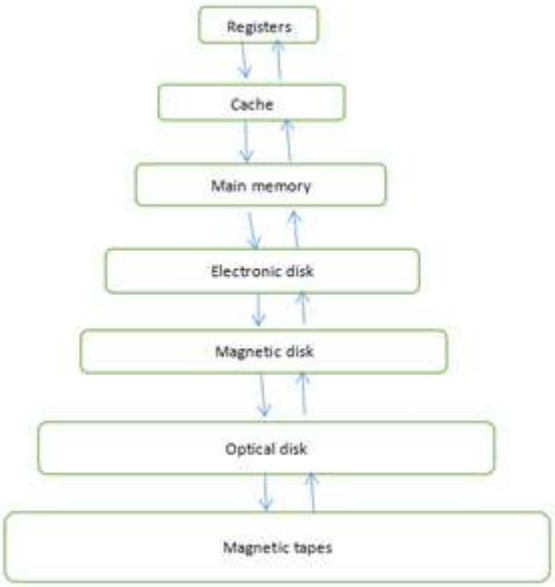


Figure 1: Memory hierarchy

What is Memory Management :

In a multiprogramming computer, the operating system resides in a part of memory and the rest is used by multiple processes. The task of subdividing the memory among different processes is called memory management. Memory management is a method in the operating system to manage operations between main memory and disk during process execution. The main aim of memory management is to achieve efficient utilization of memory.

Why Memory Management is required:

- Allocate and de-allocate memory before and after process execution.
- To keep track of used memory space by processes.
- To minimize fragmentation issues.
- To proper utilization of main memory.
- To maintain data integrity while executing of process.

Memory allocation:

To gain proper memory utilization, memory allocation must be allocated efficient manner. One of the simplest methods for allocating memory is to divide memory into several fixed-sized partitions and each partition contains exactly one process. Thus, the degree of multiprogramming is obtained by the number of partitions.

Multiple partition allocation: In this method, a process is selected from the input queue and loaded into the free partition. When the process terminates, the partition becomes available for other processes.
Fixed partition allocation: In this method, the operating system maintains a table that indicates which parts of memory are available and which are occupied by processes. Initially, all memory is available for user processes and is considered one large block of available memory. This available memory is known as “Hole”. When the process arrives and needs memory, we search for a hole that is large enough to store this process. If the requirement fulfills then we allocate memory to process, otherwise keeping the rest available to satisfy future requests. While allocating a memory sometimes dynamic storage allocation problems occur, which concerns how to satisfy a request of size n from a list of free holes. There are some solutions to this problem:
First fit:-
In the first fit, the first available free hole fulfills the requirement of the process allocated.

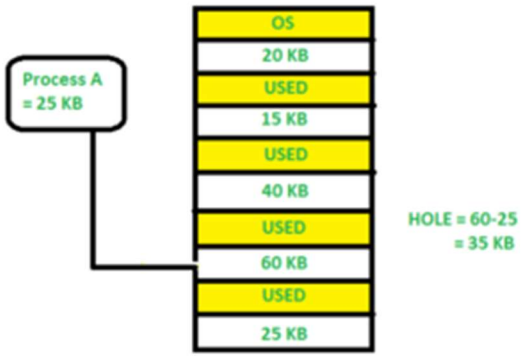
Here, in this diagram 40 KB memory block is the first available free hole that can store process A (size of 25 KB), because the first two blocks did not have sufficient memory space.

Best fit:-
In the best fit, allocate the smallest hole that is big enough to process requirements. For this, we search the entire list, unless the list is ordered by size.

Here in this example, first, we traverse the complete list and find the last hole 25KB is the best suitable hole for Process A(size 25KB).

In this method memory utilization is maximum as compared to other memory allocation techniques.

Worst fit:-In the worst fit, allocate the largest available hole to process. This method produces the largest leftover hole.



Here in this example, Process A (Size 25 KB) is allocated to the largest available memory block which is 60KB. Inefficient memory utilization is a major issue in the worst fit.

Algorithm/Flowchart:

1. **First Fit algorithm/pseudo code**

- Read all required input
- FOR i<-0 to all jobs ‘js’
  - FOR j<-0 to all blocks ‘bs’
    - IF block[j]>=jobs[i]
      - Check j^{th} block is already in use or free
        - Continue and search next free block
      - Otherwise allocate j^{th} block to i^{th} job
- Display all job with allocated blocks and fragmentation

2. First Fit algorithm/pseudo code

- Read all required input
- FOR i<-0 to all jobs 'js'
  - SET BestInd -1
  - FOR j<-0 to all blocks 'bs'
    - IF block[j]>=jobs[i]
      - IF Block is free and BestInd==-1 THEN SET BestInd j
      - ELSEIF Block is free and block[BestInd]>block[j] THEN SET BestInd j
      - ELSE continue with next block Continue and search next free block
      - IF BestInd!=-1 THEN allocate j^{th} block to i^{th} job
- Display all job with allocated blocks and fragmentation

3. Worst Fit Algorithm/Pseudo code

- Read all required input
- FOR i<-0 to all jobs 'js'
  - SET WstInd -1
  - FOR j<-0 to all blocks 'bs'
    - IF block[j]>=jobs[i]
      - IF Block is free and WstInd==-1 THEN SET WstInd j
      - ELSEIF Block is free and block[WstInd]<block[j] THEN SET WstInd j
      - ELSE continue with next block
        - Continue and search next free block
      - IF WstInd!=-1 THEN allocate j^{th} block to i^{th} job
- Display all job with allocated blocks and fragmentation

4. As above write algorithm of Next Fit strategies

Input:

- No. of jobs (js) & No. of blocks (bs)
- Job size of all jobs & Block size of all blocksFor Example:

js=4 bs=5
block[] = {100, 500, 200, 300, 600};
jobs[] = {212, 417, 112, 426};

Output:
Sample output of Worst Fit algorithm (same way generate o/p forother algorithms)-

Process No. Process Size Block no.

1 212 5

2 417 2

3 112 4

4 426 Not allocated.

Software Requirement:
1. Eclipse IDE
2. Java

Hardware Requirement:

Not specific

Conclusion: Successfully implemented simulation of memory placement strategies.

Assignment No.: 07

Problem Statement:

Write a program to simulate page replacement algorithm.

Objectives:

1. To study page replacement policies to understand memory management.
2. To understand efficient frame management usingreplacement policies.

Theory:

In an operating system that uses paging for memory management, a page replacement algorithm is needed to decide which page needs to be replaced when a new page comes in.

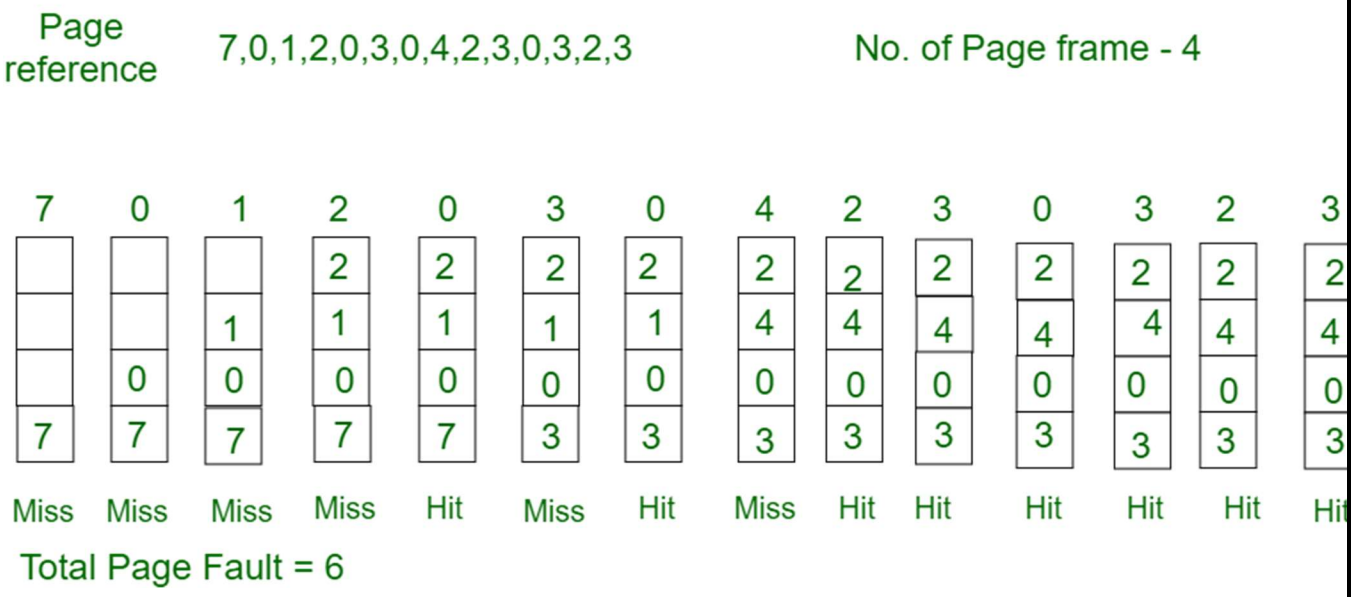
Page Fault: A page fault happens when a running program accesses a memory page that is mapped into the virtual address space but not loaded in physical memory. Since actual physical memory is much smaller than virtual memory, page faults happen. In case of a page fault, Operating System might have to replace one of the existing pages with the newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce the number of page faults.

Page Replacement Algorithms:

1. First In First Out (FIFO): This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.
Example 1: Consider page reference string 1, 3, 0, 3, 5, 6, 3 with 3 page frames. Find the number of page faults.

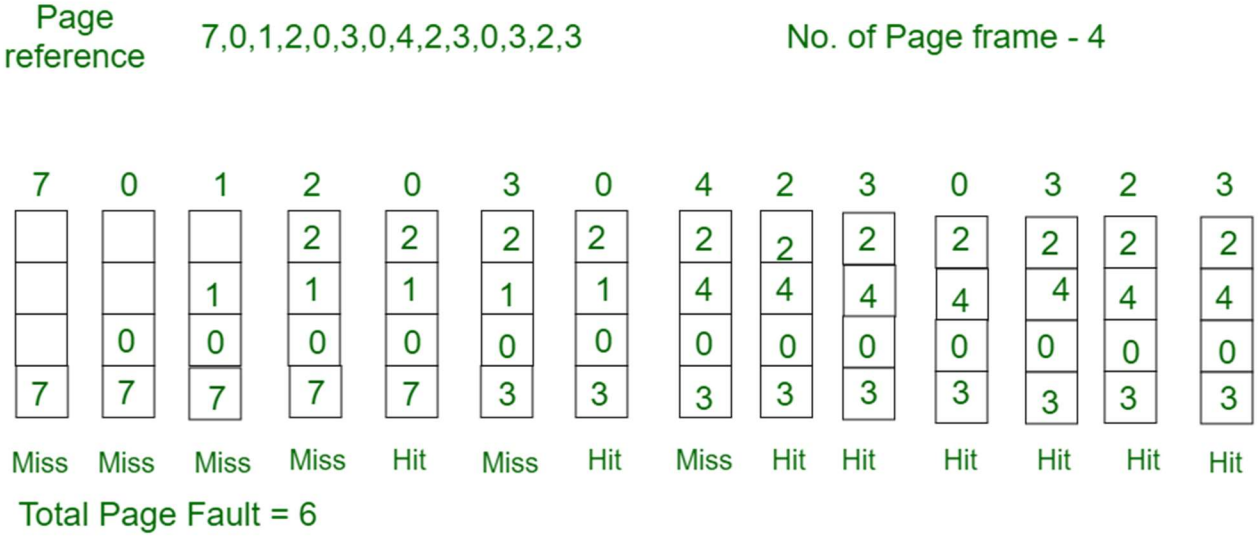
Initially, all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots —> **3 Page Faults.**
when 3 comes, it is already in memory so —> **0 Page Faults.**
Then 5 comes, it is not available in memory so it replaces the oldest page slot i.e 1. —>**1 Page Fault**
6 comes, it is also not available in memory so it replaces the oldest page slot i.e 3 —>**1 Page Fault.**
Finally, when 3 come it is not available so it replaces 0 **1 page fault.**
Belady's anomaly proves that it is possible to have more page faults when increasing the number of page frames while using the First in First Out (FIFO) page replacement algorithm.
For example, if we consider reference strings 3, 2, 1, 0, 3, 2, 4, 3, 2, 1, 0, 4, and 3 slots, we get 9 total page faults, but if we increase slots to 4, we get 10-page faults.

2. Optimal Page replacement: In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.
Example-2: Consider the page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3 with 4 page frame. Find number of page fault.



Initially, all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults** 0 is already there so —> **0 Page fault.** when 3 came it will take the place of 7 because it is not used for the longest duration of time in the future.—>**1 Page fault.** 0 is already there so —> **0 Page fault.** 4 will takes place of 1 —> **1 Page Fault.**
Now for the further page reference string —> **0 Page fault** because they are already available in the memory.
Optimal page replacement is perfect, but not possible in practice as the operating system cannot know future requests. The use of Optimal Page replacement is to set up a benchmark so that other replacement algorithms can be analyzed against it.

3. Least Recently Used: In this algorithm, page will be replaced which is least recently used.
Example-3: Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3 with 4 page frames. Find number of page faults.



Here LRU has same number of page fault as optimal but it may differ according to question.

Initially, all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**
0 is already their so —> **0 Page fault.** when 3 came it will take the place of 7 because it is least recently used —>**1 Page fault**

0 is already in memory so —> **0 Page fault**.
4 will takes place of 1 —> **1 Page Fault**
Now for the further page reference string —> **0 Page fault** because they are already available in the memory.
4. Most Recently Used (MRU): In this algorithm, page will be replaced which has been used recently. Belady's anomaly can occur in this algorithm.

Algorithm/Flowchart:

FIFO :

- Start the process
- Read number of pages n
- Read number of pages no
- Read page numbers into an array a[i]
- Initialize avail[i]=0 .to check page hit
- Replace the page with circular queue, while re-placing check page availability in theframe Place avail[i]=1 if page is placed in the frame Count page faults
- Print the results.
- Stop the process.

LEAST RECENTLY USED

- Start the process
- Declare the size
- Get the number of pages to be inserted
- Get the value
- Declare counter and stack
- Select the least recently used page by counter value
- Stack them according the selection.
- Display the values
- Stop the process

OPTIMALALGORTHIM:

- Start Program
- Read Number Of Pages And Frames
- Read Each Page Value
- Search For Page In The Frames
- If Not Available Allocate Free Frame
- If No Frames Is Free Replace The Page With The Page That Is LeastUsed 7.Print Page Number Of Page Faults
- Stop process.

Input:
Number of frames.
Number of pages.
Page sequence

Output:
1. Sequence of allocation of pages in frames (for each algorithm)
2. Cache hit and cache miss ratio.
3.

Software Requirement:
1. Fedora
2. Eclipse.
3. JDK

Hardware Requirement:
No Specific

Conclusion: Successfully implemented all page replacement policies.

Part II : Elective I

Internet of Things and Embedded Systems

Assignment No.: 01

Problem Statements:

Understanding the connectivity of Raspberry-Pi / Adriano with IR sensor. Write an application to detect obstacle and notify user using LEDs.

Objectives:

1. To understand the concept of Proximity sensor.
2. To interface Proximity sensor with Raspberry Pi model.
3. To program the Raspberry Pi model to detect the nearest object using proximity sensor and give indication through led.

Theory:

Sensor:

1. A Sensor or a Detector is a device that is used to convert a physical parameter into a signal that can be measured or monitored.
2. For example a GAS Sensor monitors gas concentration (PPM) and converts it into electrical signal that can be measured.
3. The input parameter can be different like Light, Temperature, Humidity pressure etc but the output is generally a human readable or electrically monitorable.
4. According to Oxford Dictionaries definition of Sensor is "A device which detects or measures a physical property and records, indicates, or otherwise responds to it."
5. Meaning of Sensor as per Wikipedia "A sensor is an object whose purpose is to detect events or changes in its environment, and then provide a corresponding output"

Types of Sensors:

A sensor is classified based on various aspects such as –

1. Application Based: Industrial Sensor, Automotive Sensor etc
2. Output Based : Resistive output, Differential Output, Differential output, voltage output etc
3. Parameter Sensing Based: Light, Temperature etc

Commanly used Sensors:

Light Sensor	Bio-Medical Sensor	Compass Sensor
Temperature Sensor	Sound Sensor	Flow Sensor
Proximity Sensor	Tilt Sensor	Humidity Sensor
Pressure Sensor	Hall effect Sensor	Level Sensor
GAS Sensor	Accelerometer Sensor	Motion Sensor
Current Sensor	RPM Sensor, Force Sensor	Speed Sensor

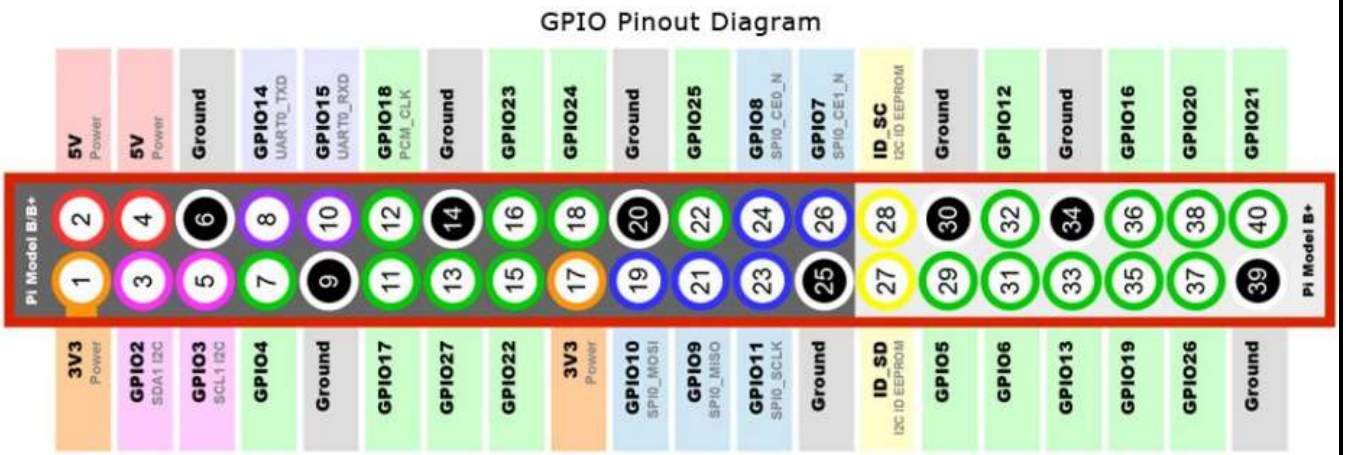
Actuators:

- An actuator is a component of a machine that is responsible for moving or controlling a mechanism or system.
- An actuator requires a control signal and a source of energy. The control signal is relatively low energy and may be electric voltage or current, pneumatic or hydraulic pressure, or even human power.
- When the control signal is received, the actuator responds by converting the energy into mechanical motion.
- Following basic actuators are used for signaling and output purpose:

– LED
– RGB LED
– Buzzer
– Servo Motor
– DC Motor
– Relay

Sensors:

GPIO Pins:

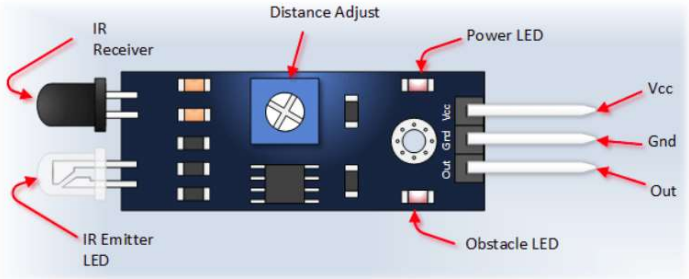


GPIO Modes:

- The GPIO.BOARD option specifies that you are referring to the pins by the number of the pin the the plug - i.e the numbers printed on the board .
- The GPIO.BCM option means that you are referring to the pins by the "Broadcom SOC channel" number.
- Unfortunately the BCM numbers changed between versions of the Pi1 Model B. – The Model B+ uses the same numbering as the Model B r2.0, and adds new pins (board numbers 27-40). – The Raspberry Pi Zero, Pi 2B and Pi 3B use the same numbering as the B+.

IR Sensors:

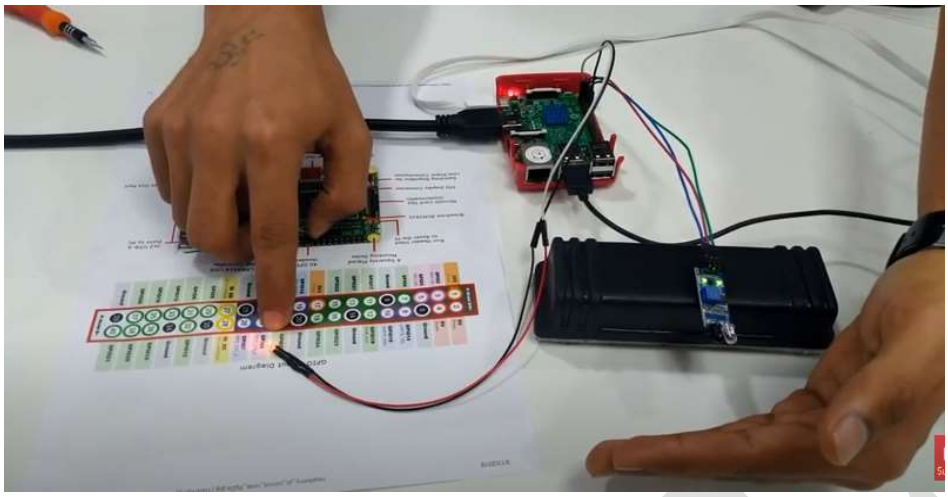
- An infrared sensor is an electronic instrument which is used to sense certain characteristics of its surroundings by either emitting and/or detecting infrared radiation.
- Infrared sensors are also capable of measuring the heat being emitted by an object and detecting motion.
- Infrared waves are not visible to the human eye. In the electromagnetic spectrum, infrared radiation can be found between the visible and microwave regions.
- The infrared waves typically have wavelengths between 0.75 and 1000µm.



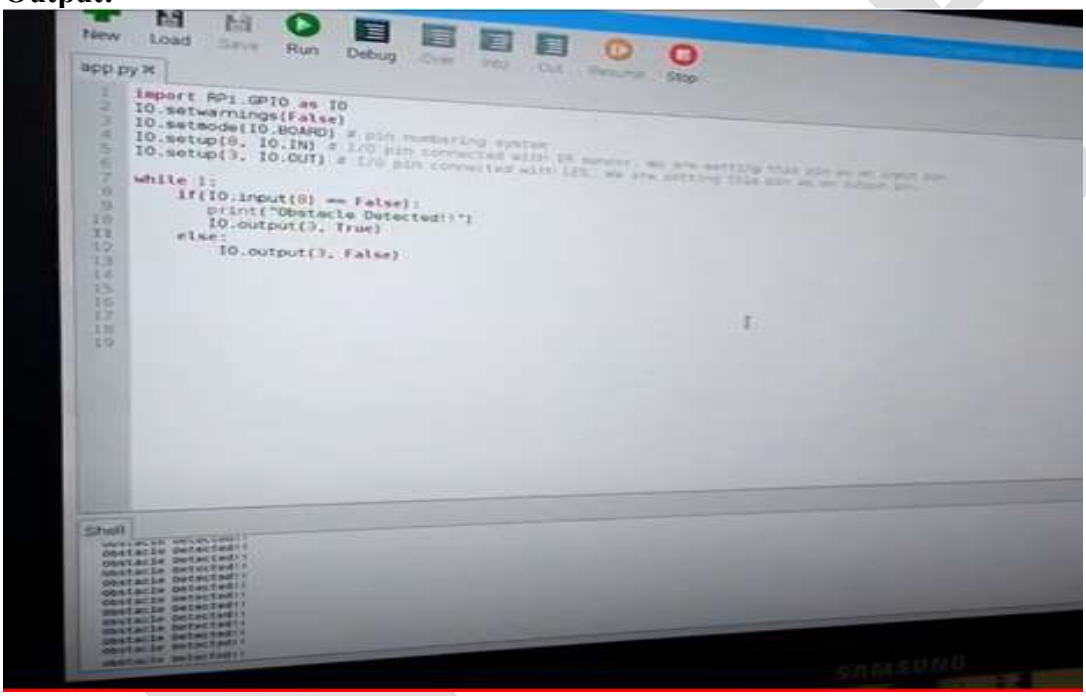
Algorithm/ Flow Chart:

1. Import GPIO and Time library
2. Set mode i.e. GPIO.BOARD
3. Set GPIO pin ‘15’ as Input
4. Set GPIO pin ‘16’ as Input
5. Read input from GPIO pin ‘15’
6. Store the input value in the variable ‘i’
7. If (i==1) then print the message as “Object is detected” and make the LED ON
8. If (i==0) then print the message as “No object detected” and make the LED OFF

Input:



Output:



Software Requirement:
Raspbian OS (IDLE)

Hardware Requirement:
Raspberry Pi Board
Proximity sensor, Led, 330 ohm register
Monitor

Conclusion:
Successfully done the connectivity of Raspberry-Pi / Adriano with IR sensor. Tested connectivity by using LEDs to detect obstacle.

Assignment No.: 02

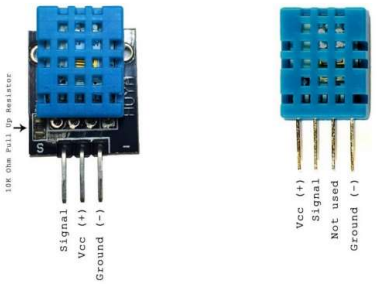
Problem Statements:
Understanding the connectivity of Raspberry-Pi /Beagle board circuit with temperature sensor. Write an application to read the environment temperature. If temperature crosses a threshold value, generate alerts using LEDs.

Objectives:

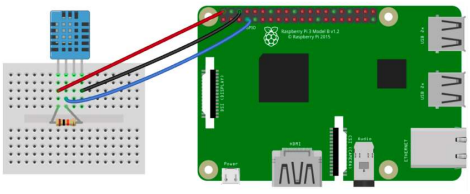
1. To understand the concept of Temperature-Humidity sensor (DHT11).
2. To interface Temperature-Humidity sensor with Raspberry Pi model.
3. To program the Raspberry Pi model to measure the real time Temperature and Humidity of the Environment.

Theory:
Temperature Sensor – DHT11

- The DHT11 temperature and humidity sensor is a nice little module that provides digital temperature and humidity readings.
- It's really easy to set up, and only requires one wire for the data signal.
- These sensors are frequently used in remote weather stations, soil monitors, and home environment control systems.
- The programming is simple too, and many libraries and example code in both Python and C already exist.
- The DHT11 contains a surface mounted NTC thermistor and a resistive humidity sensor.
- An IC on the back of the module converts the resistance measurements from the thermistor and humidity sensor into digital outputs of degrees Celsius and Relative Humidity.
- There are two variants of the DHT11 you're likely to come across. One is a four pin stand alone module, and the other is a three pin, PCB mounted module. The pinout is different for each one, so connect the DHT11 according to which one you have:



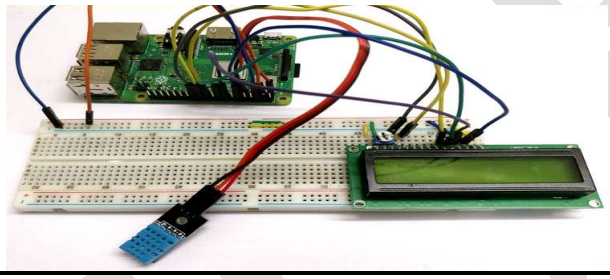
Temperature Sensor connectivity with Raspberry Pi.



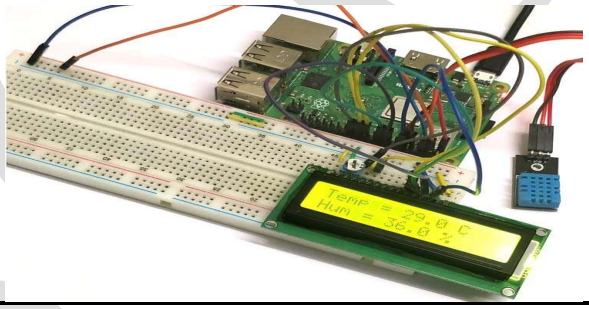
Algorithm/ Flow Chart:

1. Import GPIO, time and dht11 libraries.
2. Set all the warnings as False.
3. Set mode i.e. GPIO.BOARD
4. Read data using GPIO pin number 7 (dhtPin)
5. Write 'while loop' for displaying Temperature and Humidity values continuously
6. First Read the GPIO pin and Store the data in dhtValue Variable.
7. Print the temperature value.
8. Print the Humidity value.
9. Give delay of 1 second

Input:



Output:



Software Requirement:
Raspbian OS (IDLE)

Hardware Requirement:

1. Raspberry Pi Board module.
2. Temperature-Humidity sensor (DHT11) module.
3. Monitor

Conclusion:

1. Successfully done the connectivity of Raspberry-Pi /Beagle board circuit with temperature sensor.
2. Successfully implemented an application to read the environment temperature. If temperature crosses a threshold value, generate alerts using LEDs.

Assignment No.: 03

Problem Statements:
Understanding and connectivity of Raspberry-Pi /Beagle board with camera. Write an application to capture and store the image.

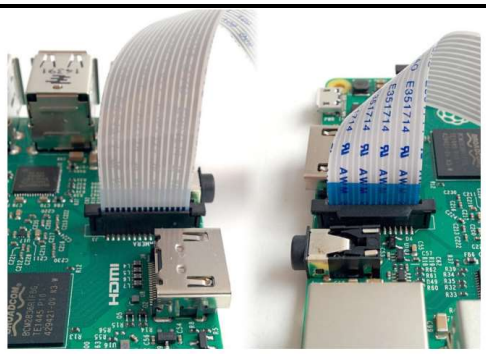
Objectives:

1. To understand the working of Raspberry Pi Camera .
2. To interface Raspberry Pi Camera with Raspberry Pi model.
3. To program the Raspberry Pi model to control the Raspberry Pi Camera Preview.
4. To program the Raspberry Pi model to capture still images from the Raspberry Pi Camera

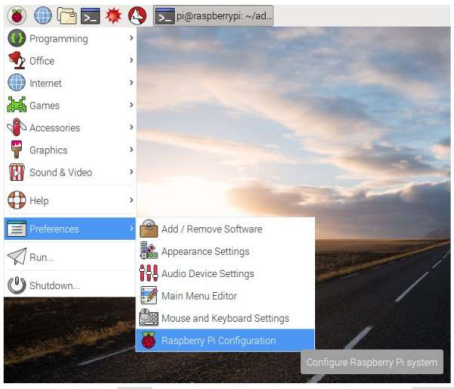
Theory:
Raspberry Pi Camera Module
The Raspberry Pi Camera Module v2 replaced the original Camera Module in April 2016. • The v2 Camera Module has a Sony IMX219 8- megapixel sensor (compared to the 5-megapixel OmniVision OV5647 sensor of the original camera). • The Camera Module can be used to take highdefinition video, as well as stills photographs. It's easy to use for beginners, but has plenty to offer advanced users if you're looking to expand your knowledge. • You can also use the libraries we bundle with the camera to create effects.
You can read all the gory details about IMX219 and the Exmor R back-illuminated sensor architecture on Sony's website, but suffice to say this is more than just a resolution upgrade: it's a leap forward in image quality, colour fidelity, and low-light performance. • It supports 1080p30, 720p60 and VGA90 video modes, as well as still capture. It attaches via a 15cm ribbon cable to the CSI port on the Raspberry Pi.
The camera works with all models of Raspberry Pi 1, 2, and 3. • It can be accessed through the MMAL and V4L APIs, and there are numerous thirdparty libraries built for it, including the Picamera Python library. • The camera module is very popular in home security applications, and in wildlife camera traps.

Pi Camera:

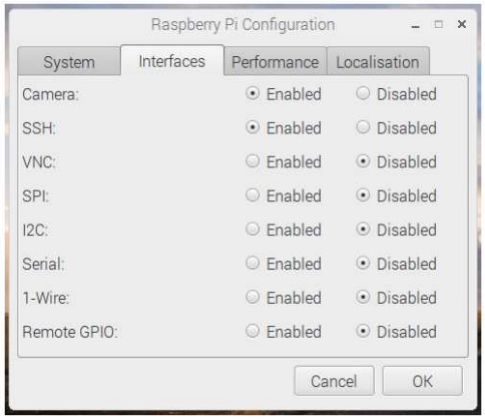
Connectivity of Camera with Raspberry Pi



Open Raspberry Pi Configuration



Enable the camera interface:



Algorithm/ Flow Chart:

a. To program the Raspberry Pi model to control the Raspberry Pi Camera Preview:

1. Import Picamera library o Import time library o Create a variable(instance) of PiCamera class

2. Display the camera preview on screen using the command “start_preview()”.

3. We can define 10 second delay to see the camera preview.

4. To stop camera preview after 10 second, use the command “stop_preview()”.

b. To program the Raspberry Pi model to capture still images from the Raspberry Pi Camera 34.

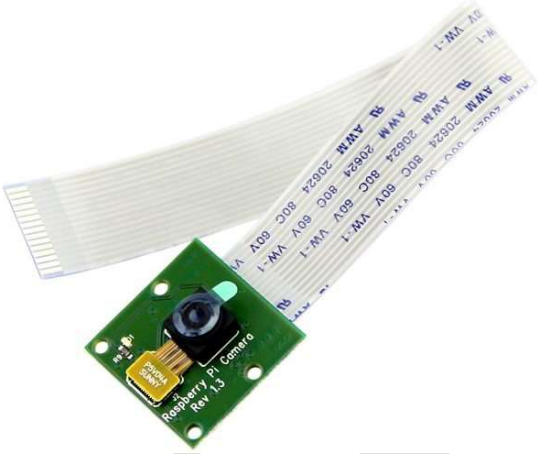
5. Import picamera library o Import time library.

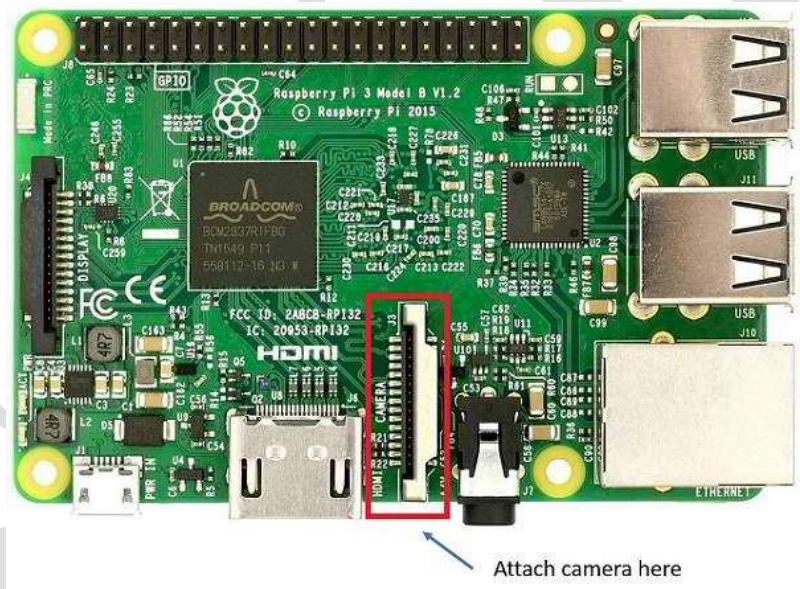
6. Create a variable(instance) of PiCamera class.

7. Display the camera preview on screen using start_preview().

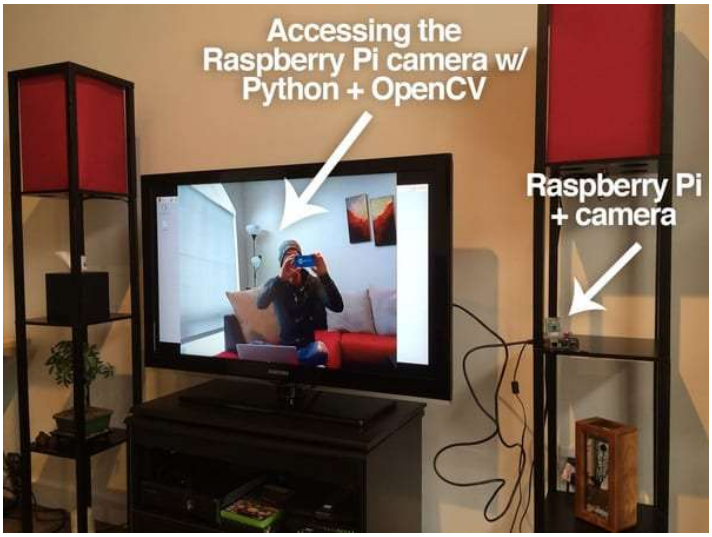
8. We can define 5 second delay to see the camera preview.
9. Capture the image using camera.capture('path of the image. extension').
10. Then stop the camera preview using the command "stop_preview()".

Input:





Output:



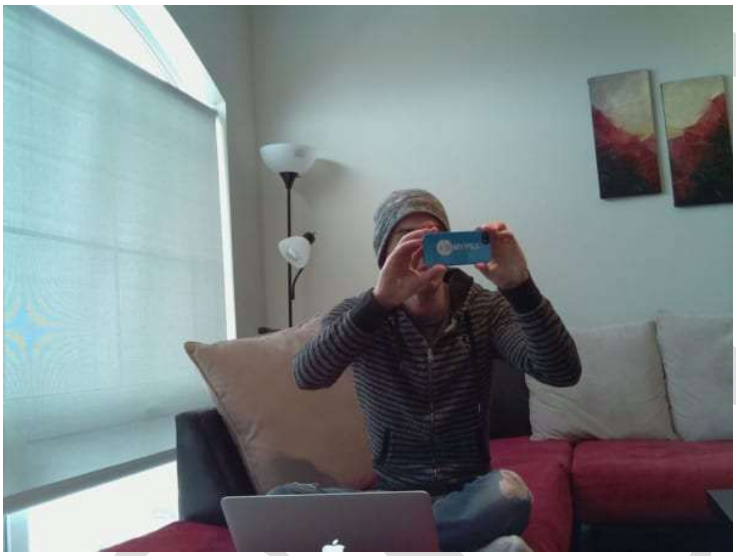


Image Captured by using Raspberry-pi Camera.

Software Requirement:

1. Raspbian OS
2. IDLE IDE

Hardware Requirement:

1. Raspberry Pi Board module.
2. Pi-Camera module.
3. Monitor

Conclusion:

1. Successfully done the connectivity of Raspberry-Pi /Beagle board with camera.
2. Successfully implemented the application to capture and store the image.

Assignment No.: 04

Problem Statements:
Create a small dashboard application to be deployed on cloud. Different publisher devices can publish their information and interested application can subscribe.

Objectives:

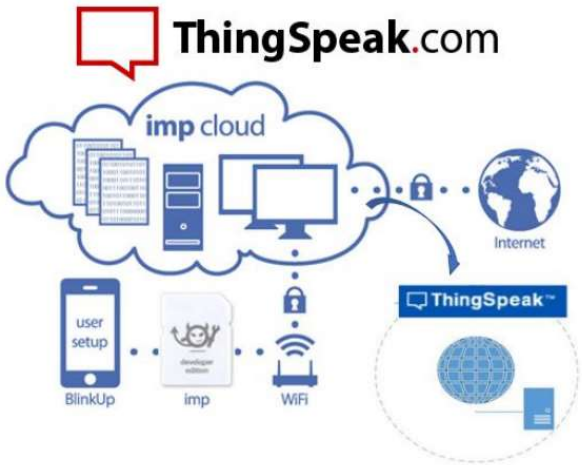
1. To understand creation of a small application to deployed on cloud.
2. To understand steps required to publish information on cloud using ThingSpeak

Theory:
IOT Platform:
The IoT platforms are suites of components those help to setup and manage the internet connected devices. • A person can remotely collect data, monitor and manage all internet connected devices from a single system. • There are a bunch of IoT platforms available online but building an IoT solution for a company is all depend on IoT platform host and support quality.

IOT Cloud Platform:

- Kaa IoT Platform
- SiteWhere: Open Platform for the Internet of Things
- ThingSpeak: An open IoT platform with MATLAB analytics
- DeviceHive: IoT Made Easy
- Zetta: API-First Internet of Things Platform
- DSA: Open Source Platform & "Toolkit" for Internet Of Things Devices
- Thingsboard.io Open-source IoT Platform
- Thinger.io: The Opensource Platform for Internet of things
- WSo2- Open source platform for Internet of Things and mobile projects

ThinkSpeak:

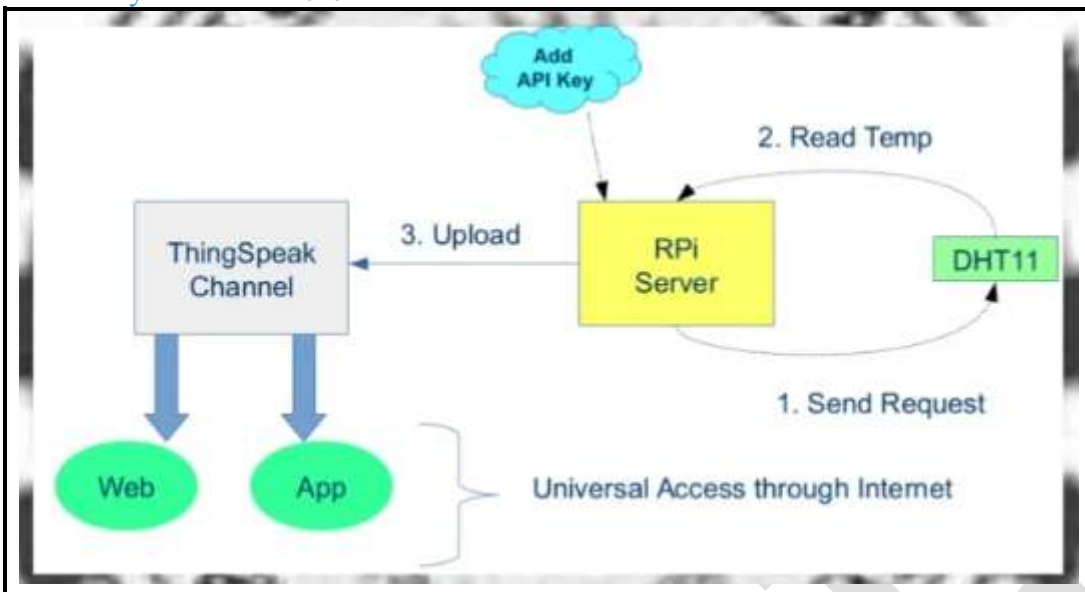


ThinkSpeak Features:

- Collect data in private channels
- Share data with public channels
- RESTful and MQTT APIs
- MATLAB analytics and visualizations
- Alerts
- Event scheduling
- App integrations
- Worldwide community

Algorithm/ Flow Chart:
In this Practical we are going to upload sensed temperature value on ThingSpeak.

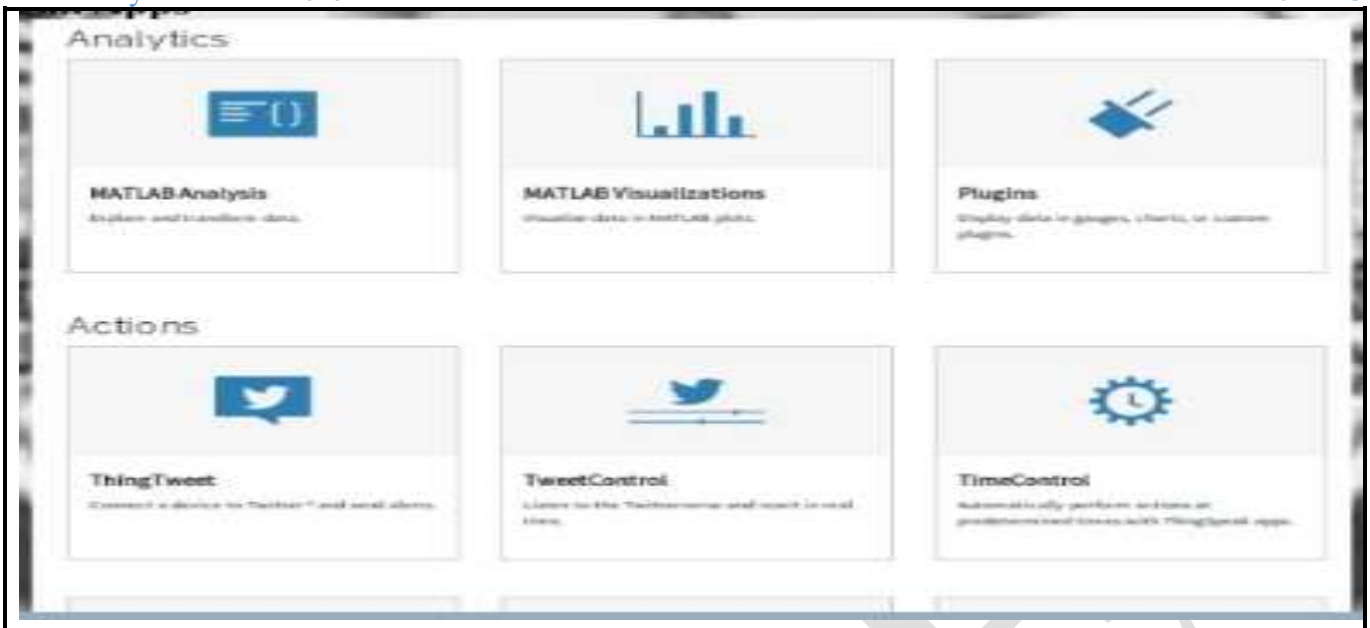
```
import httplib,urllib
import time,Adafruit_DHT
key='8QPPQALZTUZUZ7IS'
while True:
   h,t=Adafruit_DHT.read_retry(11,4)
   print "temp:",t
   param=urllib.urlencode({'field1':t,'key':key})
   headers={"content-typZZe":"application/x-www-form-urlencoded","Accept":"text/plain"}
   conn=httplib.HTTPConnection("api.thingspeak.com:80")
   try:
      conn.request("POST","/update",param,headers)
      response=conn.getresponse()
      print response.status,response.reason
      data=response.read()
      conn.close()
   except:
      print "connection Failed"
```



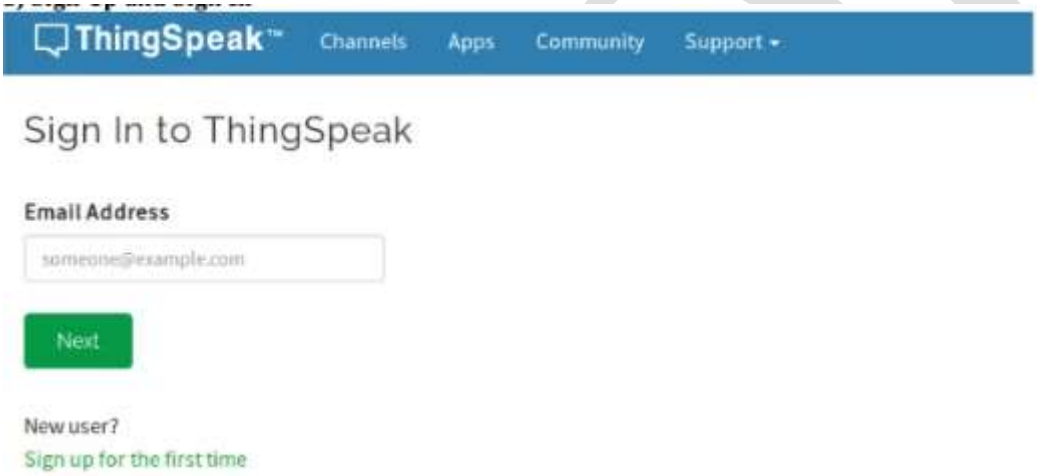
Steps for configuration of ThingSpeak:

4. **Open ThingSpeak Link**

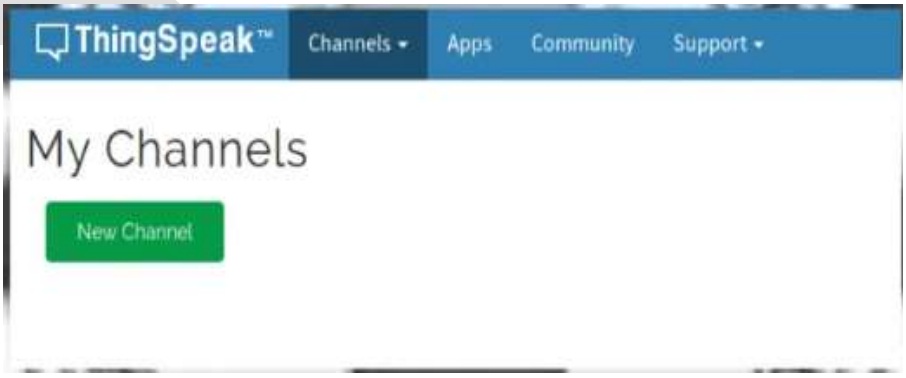
5. **ThingSpeak APP**



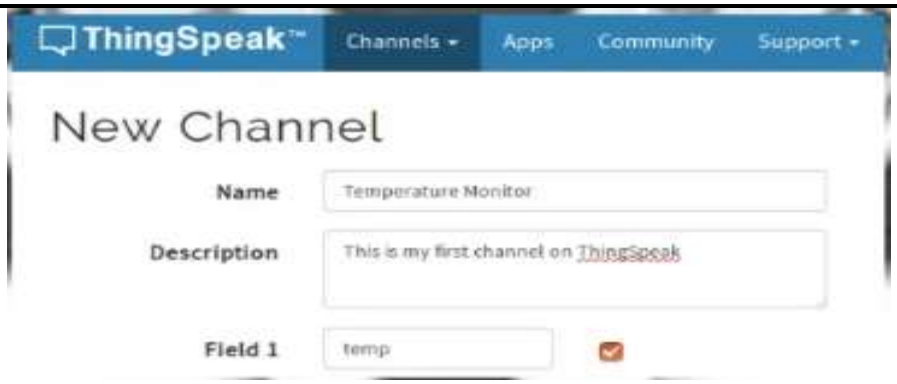
3.Sign Up and Sign In



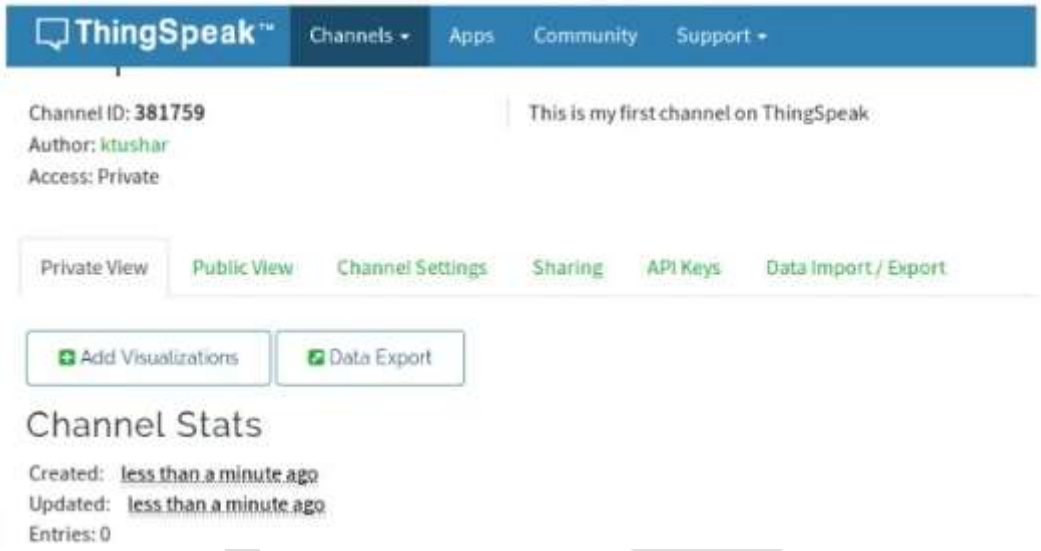
4. After Sign In



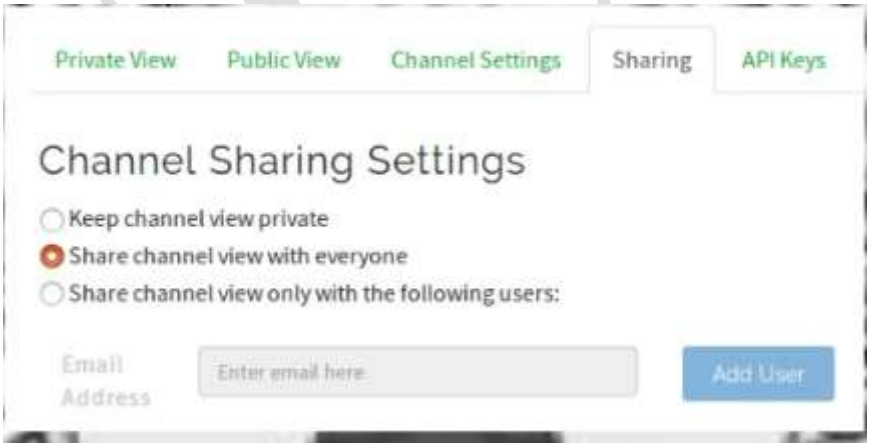
5. Create New Channel



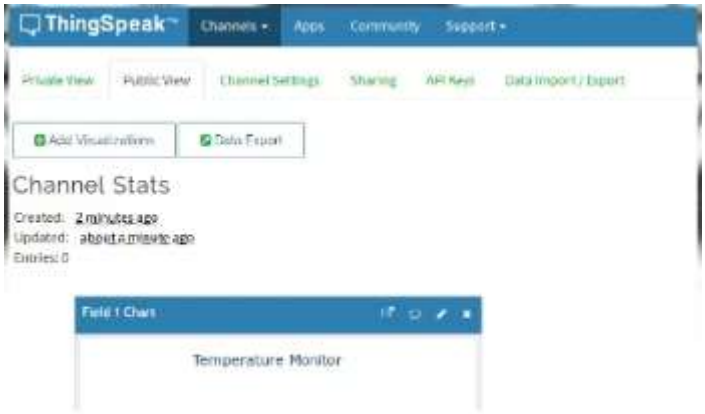
6. **First Time Channel View**



7. **Make Our Channel Public**



8. **Now public view will be available**



9. API key for R/W Operation

Input:
Follow all above steps to deploy application on cloud.

Output:
Publisher can see their information on cloud.

Software Requirement:

1. Raspbian OS-IDLE
2. Account at ThingSpeak

Hardware Requirement:

1. Raspberry Pi Board Module.
2. DHT-11 Sensor

Conclusion:
Successfully done deployment of application on cloud.

Part II: Elective I

Human Computer Interface

Assignment No.: 01

Problem Statements:
Design a paper prototype for selected Graphical User Interface.

Objectives:
1. To study how to design paper prototype for selected graphical user interface.
2. To study the paper prototyping design for different types of design purpose.
3. To study What is Paper prototyping?

Theory:
Paper Prototyping:

- Paper prototyping is a process where design teams create paper representations of digital products to help them realize concepts and test designs.
- They draw sketches or adapt printed materials and use these low-fidelity screenshot samples to cheaply guide their designs and study users' reactions from early in projects.
- "Paper prototyping is great for exploring design possibilities. You can try as many as you want, and if they don't work for you it's fine, just throw them in the bin and start over.
- It opens your eyes on things you haven't thought of and gives you new design perspectives."
- Paper prototyping is a core activity in design processes.
- You depict screenshots (in what you can call "paper-shots") to help determine how your design/product should appear.
- Like other forms of low-fidelity prototyping—e.g., card sorting—paper prototyping is a cheap-and-easy way to help shape concepts. If you use it early on, you can prevent unwanted development costs.

Pros and Cons of Paper Prototyping
Consider the strengths and limitations of paper prototyping:
Pros:

1. Quick iteration:You can build overviews without getting bogged down in details. In minutes, you can see whether an idea works on paper.
2. Cheap:Paper is inexpensive; so are printed prototyping materials/kits.
3. Universal: Everyone can make rough sketches of ideas. Stakeholders from

outside the design team can join in.

4. Pieces serve as documentation:Later on, you'll have annotated hard-copy evidence of what works and what doesn't.
5. Team-building:When team members get creative, they can bond. Everyone can get involved in drawing, cutting and pasting and forget role/department barriers.
6. Honest feedback:People comment more freely than if they must criticize polished prototypes (i.e., someone's "baby").
7. Useful throughout the design process:You can use paper prototyping to help stay flexible about revisions throughout development.

Cons:

1. Lack of realism: Whatever you draw, you can't completely mimic an interactive design. Also, users' gut reactions will differ compared with the finished product.
2. Inappropriateness in some contexts:You can't always translate users' constraints onto paper, especially regarding accessibility. You may need a sophisticated high-fidelity prototype to capture the user experience.
3. Requires in-person testing:You have a smaller pool of test users and greater risk of missing insights.
4. Lack of user control: Without an interactive design, users must give blow-by-blow accounts of their actions and thoughts. Also, you can't moderate from a distance. You must give directions about next steps, without leading users.
5. More work: You'll make digital prototypes, anyway. These may suit your concept without the need for primitive prototypes.
6. Interpret results carefully:Users can't get a real feel of the product. Positive feedback is a good indicator of how to proceed, not a guarantee.

Algorithm/ Flow Chart:

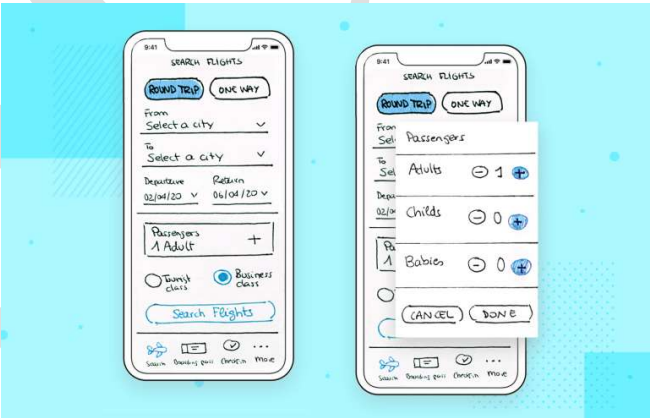
1. Gather stationery – pens, pencils, markers, paper, card, Post-its, scissors, tape, glue, rulers and suitable stencils. About $10/£10/€10 is typically enough to cover this. You can use graph paper to help guide ideas. Colored paper is great for representing buttons.

2. Get building – Just go ahead and see where you go. As you get your ideas down on paper, you can think about them more concretely. Later, you can get insights about improving them.
3. Make one sketch per screen.
4. Move quickly – If you spend too long making your prototype, you'll get attached to it. Don't waste time erasing: You want to get ideas down rather than revise them and potentially miss insights from the raw version.
5. Remember what to test – Build with that purpose in mind, but stay aware of other factors.
6. Prototype for small screens first –When you go mobile-first, you can prioritize your content better.
7. Remember the users – As you'll test your prototype against users' behaviors and needs, consider their expectations as you build.

Input:

- Paper
- Templates
- Pen or thin markers
- Ruler
- Stencils
- Scissors

Output:



Software Requirement: Not applicable

Hardware Requirement: Not applicable

Conclusion: Thus, we have studied designing of a paper prototype for selected Graphical User Interface

Assignment No.: 02

Problem Statements: Implement GOMS (Goals, Operators, Methods and Selection rules) modeling technique to model user's behavior in given scenario.

Objectives:

1. To study the design and implementation of GOMS (Goals, Operators, Methods and Selection rules) modeling technique to model user's behavior in given scenario.
2.

Theory:

What is GOMS?

– Description of the knowledge that a user must have in order to carry out tasks on a device or system.

– Representation of the "how to do it" knowledge that is required by a system in order to get the intended tasks accomplished. •

What does a GOMS task analysis involve?

– Involves defining and then describing the user's

• Goals: – Something that the user tries to accomplish (action-object pair, e.g. delete word) – Include context

• Methods: – Well learned sequence of steps that accomplish a task – How do you do it on this system? (could be long and tedious…)

• Selection Rules: – Only when there are clear multiple methods for the same goal.

• Operators: – Elementary perceptual, cognitive and motor acts that cause change (external vs. mental) – Also uses action-object pair (e.g. press key, select menu, make gesture, speak command...) – mostly defined by hardware and lower-level software.

GOMS Example:

File & directory operations:

– Delete a file, move a file, delete a directory, move a directory.

GOMS analysis

– File & directory operations: – Method for goal: delete a file.

• Step 1. drag file to trash.

• Step 2. Return with goal accomplished.

– Method for goal: move a file.

• Step 1. drag file to destination.

• Step 2. Return with goal accomplished.

– Method for goal: delete a directory.

• Step 1. drag directory to trash.

• Step 2. Return with goal accomplished.

– Method for goal: move a directory.

• Step 1. drag directory to destination.

• Step 2. Return with goal accomplished.

GOMS analysis – File & directory operations - a better version:

– Method for goal: delete an object.

• Step 1. drag object to trash.

• Step 2. Return with goal accomplished.

– Method for goal: move an object.

• Step 1. drag object to destination.

• Step 2. Return with goal accomplished.

Algorithm/ Flow chart:

1. DEFINE THE USERS TOP-LEVEL GOALS.

2. GOAL DECOMPOSITION.

- Break down each top-level goal into its own subgoals.

3. DETERMINE AND DESCRIBE OPERATORS

- Find out what actions are done by the user to complete each subgoal from step 2. These are the operators.

4. DETERMINE AND DESCRIBE METHODS.

- Determine the series of operators that can be used to achieve the goal. Determine if there are multiple methods and record them all.

5. DESCRIBE SELECTION RULES.

- If more than one method is found in step 4, then the selection rules, or which method the user will typically used, should be defined for the goal.

Input: User Goal

Output:

```
GOAL: CLOSE-WINDOW
.   [select GOAL: USE-MENU-METHOD
            .   MOVE-MOUSE-TO-FILE-MENU
            .   PULL-DOWN-FILE-MENU
            .   CLICK-OVER-CLOSE-OPTION
            GOAL: USE-CTRL-W-METHOD
            .   PRESS-CONTROL-W-KEYS]

For a particular user:

   Rule 1: Select USE-MENU-METHOD unless another
           rule applies
   Rule 2: If the application is GAME,
           select CTRL-W-METHOD
```

Software Requirement: Not applicable

Hardware Requirement: Not applicable

Conclusion: Thus we have studied Implementation of GOMS (Goals, Operators, Methods and Selection rules) modeling technique to model user's behavior in given scenario.

Assignment No.: 03

Problem Statements:
Design a User Interface in Python.

Objectives:
1. To study and design user interface in python
2. To study how user interface can design in python.

Theory:

User interface is the front-end application view to which user interacts in order to use the software. The software becomes more popular if its user interface is:

- Attractive
- Simple to use
- Responsive in short time
- Clear to understand
- Consistent on all interface screens

There are two types of User Interface:

1. **Command Line Interface:** Command Line Interface provides a command prompt, where the user types the command and feeds to the system. The user needs to remember the syntax of the command and its use.
2. **Graphical User Interface:** Graphical User Interface provides the simple interactive interface to interact with the system. GUI can be a combination of both hardware and software. Using GUI, user interprets the software.

User Interface Design Process:
The analysis and design process of a user interface is iterative and can be represented by a spiral model. The analysis and design process of user interface consists of four framework activities.

1. **User, task, environmental analysis, and modeling:** Initially, the focus is based on the profile of users who will interact with the system, i.e. understanding, skill and knowledge, type of user, etc, based on the user's profile users are made into categories. From each category requirements are gathered. Based on the requirements developer understand how to develop the interface. Once all the requirements are gathered a detailed analysis is conducted. In the analysis part, the tasks that the user performs to establish the goals of the system are identified, described and elaborated. The analysis of the user environment focuses on the physical work environment. Among the questions to be asked are:

- Where will the interface be located physically?
- Will the user be sitting, standing, or performing other tasks unrelated to the interface?
- Does the interface hardware accommodate space, light, or noise constraints?
- Are there special human factors considerations driven by environmental factors?

2. **Interface Design:** The goal of this phase is to define the set of interface objects and actions i.e. Control mechanisms that enable the user to perform desired tasks. Indicate how these control mechanisms affect the system. Specify the action sequence of tasks and subtasks, also called a user scenario. Indicate the state of the system when the user performs a particular task. Always follow the three golden rules stated by Theo Mandel. Design issues such as response time, command and action structure, error handling, and help facilities are considered as the design model is refined. This phase serves as the foundation for the implementation phase.
3. **Interface construction and implementation:** The implementation activity begins with the creation of prototype (model) that enables usage scenarios to be evaluated. As iterative design process continues a User Interface toolkit that allows the creation of windows, menus, device interaction, error messages, commands, and many other elements of an interactive environment can be used for completing the construction of an interface.
4. **Interface Validation:** This phase focuses on testing the interface. The interface should be in such a way that it should be able to perform tasks correctly and it should be able to handle a variety of tasks. It should achieve all the user's requirements. It should be easy to use and easy to learn. Users should accept the interface as a useful one in their work.

A graphical user interface (GUI) is a desktop interface that allows you to communicate with computers. They carry out various activities on desktop computers, laptops, and other mobile devices. Text-Editors and other graphical user interface applications build, read, download, and erase various types of files. You can also play games such as Sudoku, Chess, and Solitaire through these apps. Google Chrome, Firefox, and Microsoft Edge are examples of graphical user interface (GUI) Internet browsers.

Python has a variety of libraries, but these four stand out, especially in terms of GUI.

- Tkinter :

- Often referred to as the go-to GUI toolkit by a majority of Python developers, **Tkinter** was created to equip modern developers with a standard interface to the Tk GUI toolkit with its Python bindings.
- In Tkinter's world, most of the **visual elements that we're familiar with are called widgets,** and each of these widgets offers a different level of customizability.
- Tkinter comes baked into current Python installers for all major operating systems and offers a host of commonly used elements that we're sure you must be familiar with. Some of those visual elements have been listed below:
  - ✓ Frame: for providing a structure to your application.
  - ✓ Buttons: used for taking input from the user
  - ✓ Check buttons: used for making selections
  - ✓ Labels: for displaying textual information
  - ✓ File Dialogs: for uploading or downloading files to/from the application
  - ✓ Canvas: provides a space for drawing/painting things like graphs and plots

- Kivy :
  - **Written with a mix of Python and Cython**, **Kivy** is an **open-source GUI framework** for building some of the most intuitive user interfaces encompassing multi-touch applications **that implement Natural User Interface(NUI).**
  - *A NUI is a kind of interface where the user naturally learns about the various interactions provided by a user interface while they're usually kept invisible.*
  - With Kivy, interface designers **can code once and deploy to multiple platforms**, while the built-in support for OpenGL ES 2 allows them to use modern and powerful graphics and design techniques.
  - The most common use of the Kivy GUI framework in the real-world **can be seen in our Android and iOS applications**. Other widespread implementations of the framework can be seen in the **user interfaces of Linux, Windows, Raspberry Pi, and Mac OS** devices.

You can easily add this framework to your environment by following the installation instructions provided on their website.

- Python QT :

  - The **PyQt package** is built around the **Qt framework,** which is a cross-platform framework used for creating a plethora of applications for various platforms. The PyQt5 package includes a detailed set of bindings for Python based on the **latest version v5** of the Qt application framework.

  - Similar to the Qt5 framework, PyQt5 is also **fully cross-platform**. By leveraging the power of PyQt5, developers can build applications for platforms like Windows, Mac, Linux, iOS, Android, and more.

  - When it comes to creating GUIs, the PyQt5 arsenal **offers the impressive QtGui and the QtDesigner module,** which provide numerous visual elements that the developer can implement with a simple drag and drop.

  - Of course, the option of creating these elements by code also exists, allowing you to create both small-scale as well as large-scale applications with ease. Python's modularity trickles down to PyQt5 in the form of extensions, giving you a lot more features than just GUI building. If you like what you see here, you can give PyQt5 a try with the command "pip install PyQt5".

- wxPython:

  - **wxPython** is essentially a Python extension module that acts as a wrapper for the wxWidgets API.

  - wxPython **allows Python developers to create native user interfaces** that add zero additional overhead to the application.

  - The cross-platform capabilities of wxPython allow deployment to platforms like Windows, Mac OS, Linux, and Unix-based systems with little to no modifications.

- The developer duo later **released Project Phoenix** as the successor to wxPython **with support for Python 3**.

- It has been built from the ground-up to offer a more cleaner Python implementation of the wxWidgets toolkit. With its addition to PyPI, the downloads have become much smoother **with the command "pip install wxPython==4.1.1".**

Algorithm/ Flow Chart:

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Input:

Not specific.

Output:

GUI application designed by user using Tkinter.

Software Requirement:

Tkinter

Hardware Requirement: Not Specific

Conclusion: Thus, we have studied designing of a User Interface in Python.

Assignment No.: 04

Problem Statements:
To redesign existing Graphical User Interface with screen complexity.

Objectives:
1. To Study and redesign existing Graphical User Interface with screen complexity.
2. To identify and Implementation/redesign existing Graphical User Interface with screen complexity.

Theory:

What is screen complexity?

Screen complexity describes how difficult it is to navigate a user interface (UI). It's influenced by factors such as size, grouping, alignment, and element repetition.

Why is reducing complexity important in UI design?

Reducing screen complexity is an essential part of UI design because a complexity issue affects everything from product adoption to customer satisfaction, churn, and revenue.

User-friendly design converts potential visitors into buyers

Minimizing complexity leads to higher revenue since a user-friendly design is more likely to convert potential visitors into buyers.

If potential customers are testing out a free trial of your product, but are disappointed with a complex interface, they'll quit trying and won't upgrade.

It's not just limited to the in-app experience either. Build a user-friendly website and a user-friendly customer support experience to improve your conversion rates even more.

Simplifying design means more engagement

Simplifying design improves engagement because it reduces friction. When users can navigate through a clear product experience, they'll stay engaged for longer.

Overly complex designs are time-consuming and frustrating to navigate. In this digital age, customers have high expectations and minimal patience for any friction. They'll move on *fast* if you can't provide a seamless experience.

Consistent experience that drives retention and loyalty

In the long term, delivering a more consistent experience will keep customers around for longer and improve loyalty – both of which contribute to higher customer lifetime value (LTV), reduced churn, and more word-of-mouth growth.

If customers consistently have a positive experience with your product, both specifically with the UI design and the experience as a whole, they'll keep their subscription and spread the word to their network.

Algorithm/ Flow Chart:

Method for Measuring Complexity:

1. Draw a rectangle around each element on a screen, including captions, controls, headings, data, title, and so on.
2. Count the number of elements and horizontal alignment points (the number of columns in which a field, inscribed by a rectangle, starts).
3. Count the number of elements and vertical alignment points (the number of rows in which an element, inscribed by a rectangle, starts).
3. Calculate number of bits required by horizontal (column) alignment points and number of bits required by vertical (row) alignment points by applying following formula for calculating the measure of complexity.

$$C = -N \sum_{n=1}^{m} p_n \log_2 p_n$$

C , complexity of the system in bits

N , total number of events (widths or heights)

m, number of event classes (number of unique widths or heights)

pn , probability of occurrence of the nth event class (based on the frequency of events within that class)

5. Calculate overall complexity by adding the number bits required by horizontal

alignment points and vertical alignment points.

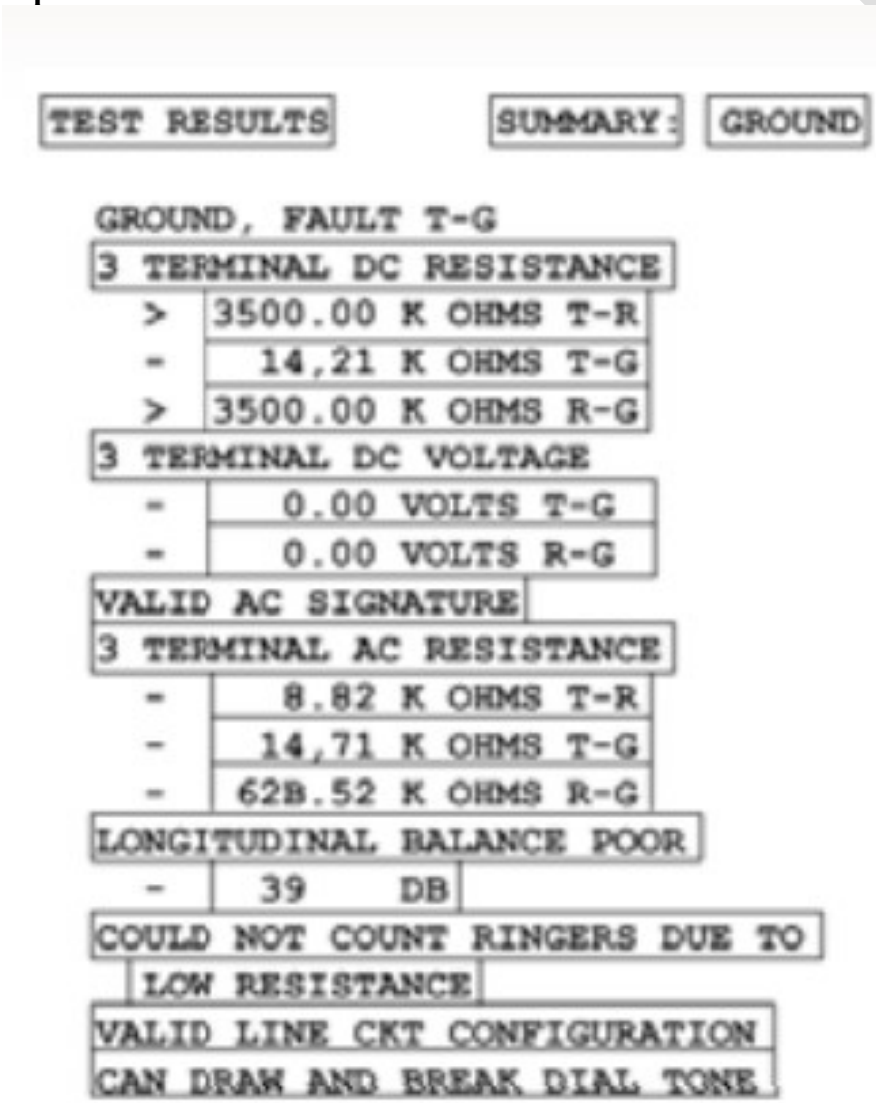
Guidelines for reducing complexity:

1. The way to minimize screen complexity is to reduce the number of controls displayed. Fewer controls will yield lower complexity measures.

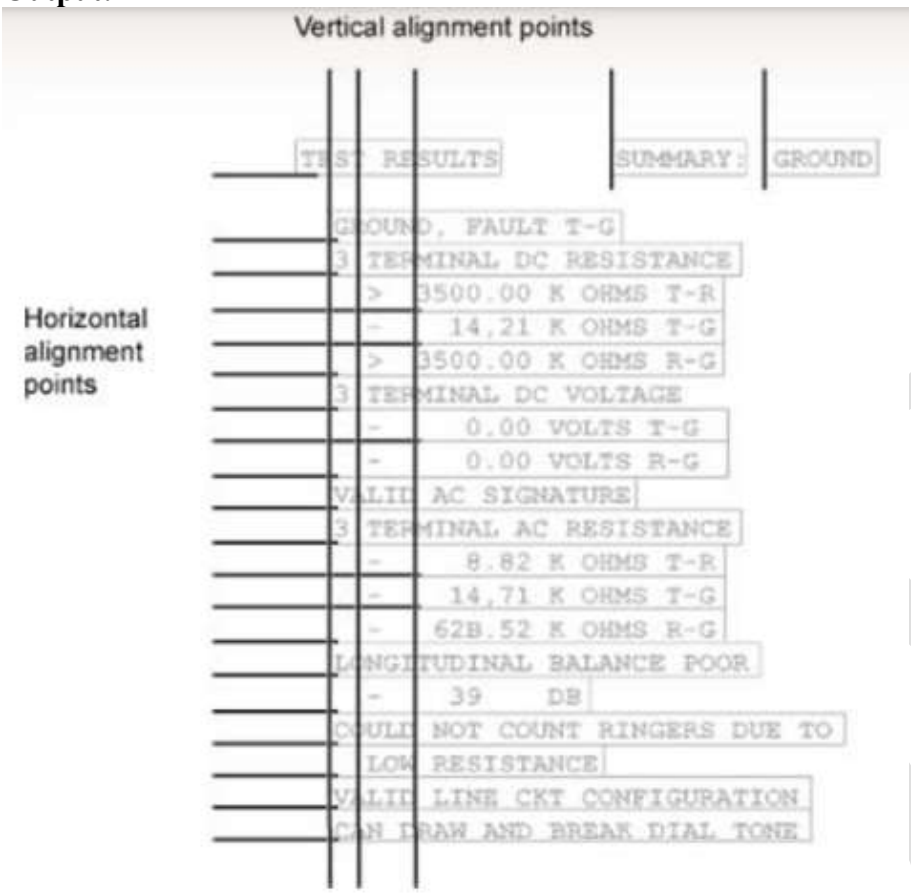
2. Optimize the number of elements on a screen, within limits of clarity.

. Alignment: Minimize the alignment points, especially horizontal or columnar. Fewer alignment points will have a strong positive influence on the complexity calculation. When things don't align, a sense of clutter and disorganization often results. In addition to reducing complexity, alignment helps create balance, regularity, sequentially, and unity.

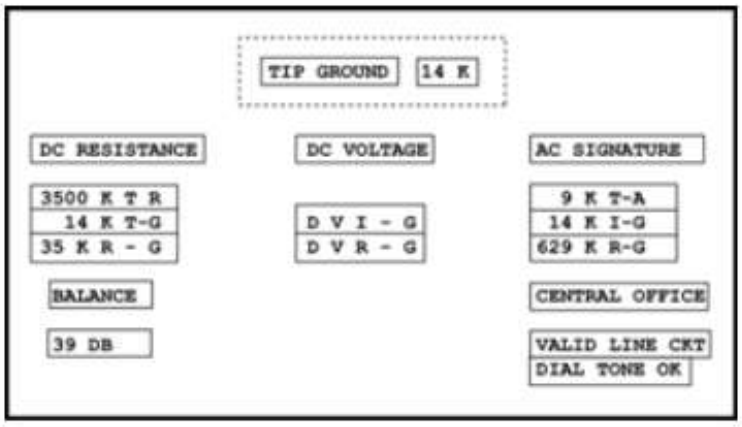
Input:



Output:



Redesigned screen



Software Requirement:

Not Specific

Hardware Requirement:

Not Specific

Conclusion: Thus, we have studied redesigning of Graphical User Interface.