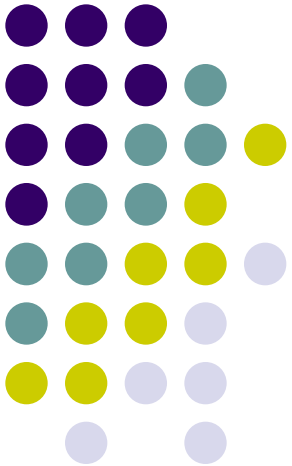
Pdf Converter

(earlier Pdf2Doc)

CS 293 Final Project Demo

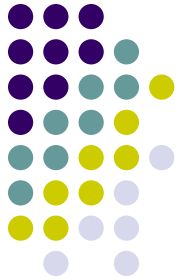
24th Nov 2012



Shubham Mehta, 110050013

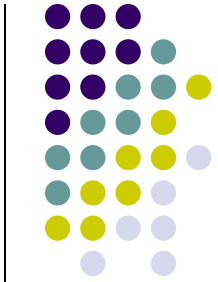
Rahul Singhal, 110050023

Outline <for a total of 30 mins>



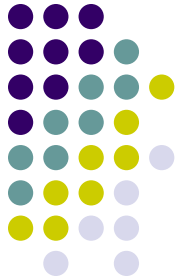
- Aim of project (1 mins)
- Demo (5 mins)
- Teamwork Details (0.5 min)
- Design Details –Algorithm (5 mins)
- Design Details – Implementation (8 mins)
- Viva (9 mins)
- Transition time to next team (2 mins)

Aim of the project



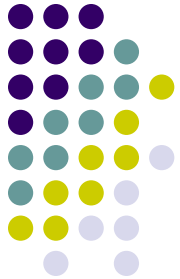
- A Program that converts .pdf file into .tex and .html file format preserving its basic structure.
- Analysis of algo and data structures used
  - *We have made our own parser in which we have identified font, alignment of text, type and level of bullets, starting of section, image position and text position to make it look like a table in case we have detected a table*.
  - *We have mainly manipulated strings using stacks, vectors and unordered maps. Stacks have been mainly used to keep track of starting and ending of bullets and sections and to store Latex syntex required at the start and end of a bullet, sections and alignments.*

Demo



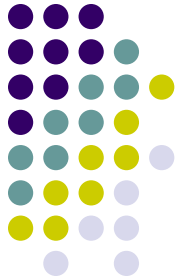
- *We have used to pdf-box library to extract text and image positions corresponding to (char) text and images present in the pdf.*
- *Then we have made two parsers one to make **.html file** from the data extracted and other to make **.tex file** from the data extracted using java code.*

Demo
HTML Parser



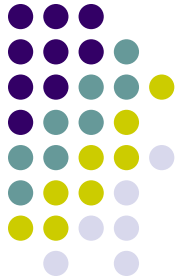
- Last Letter Class
  - *It store all the attributes corresponding to Last letter we have parsed so that we can compare its yCoord (to check newline), xCoord(to check new word), and font with new letter parsed.*
- Font Class
  - *It stores various attributes corresponding to a font like space width(needed to check new word), weight(if it is bold), style (if it is italic), font size and font family.*
  - *Operators have been overloaded in to compare different fonts encountered.*

Demo
HTML Parser



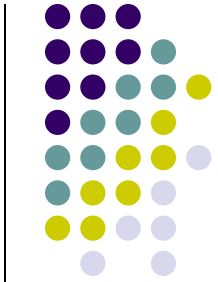
- Html_class Class
  - *It stores attributes corresponding to various classes (html styling class) encountered while text parsing like font(that has been used in that class), divNo (states the page no. on which that styling class is encountered).*
  - *Every time a new font is encountered a new styling class is made with font mapped to the class no. and syntax corresponding to that class is written in the "head .txt" file (contains syntax that has to be included in final.html at the top describing properties of various styling classes).*

Demo
HTML Parser



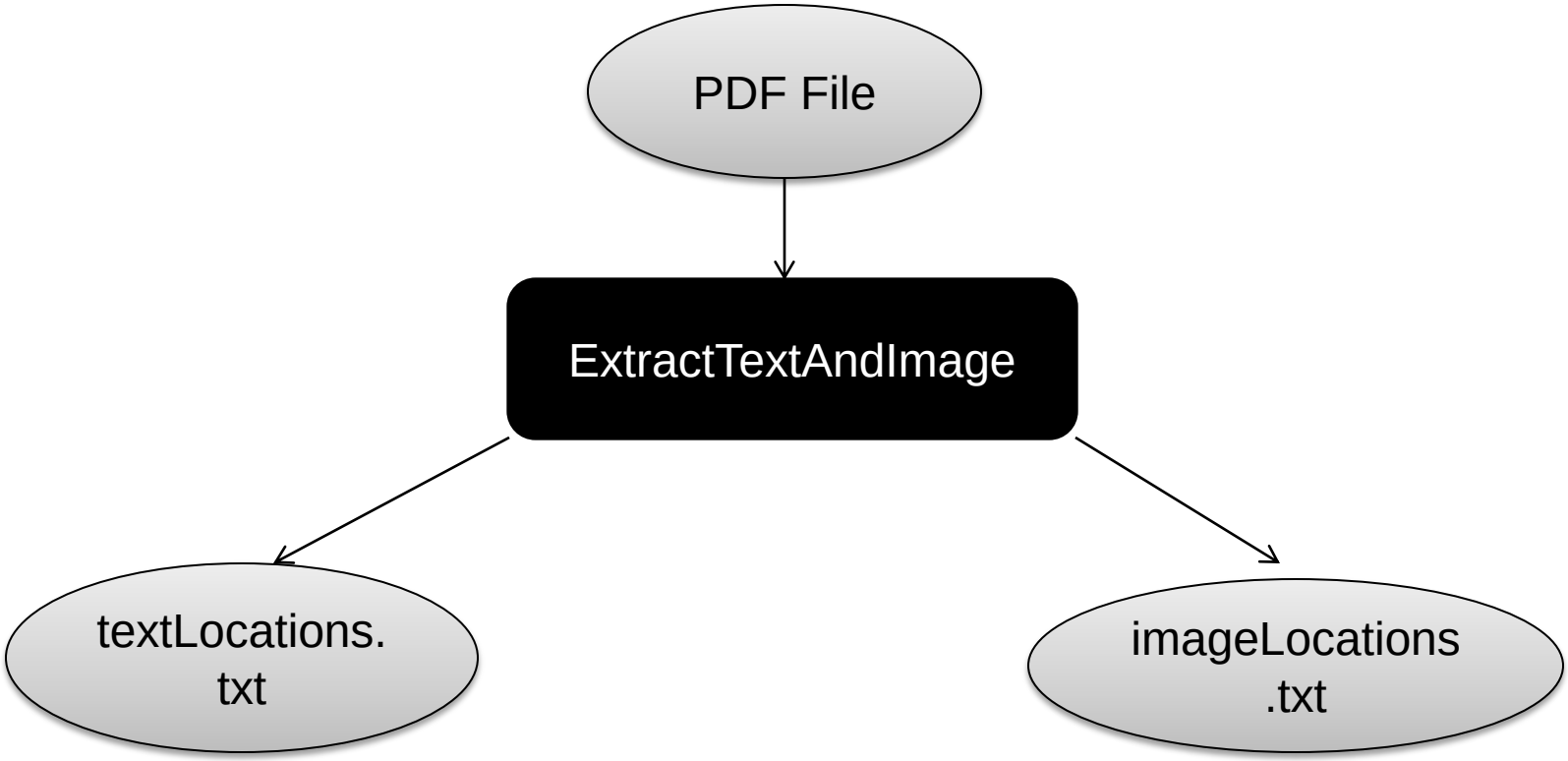
- Parser Class
  - *It parses all the text extracted from the pdf which is stored in “textLocations.txt” and detects whether a new word has started (using xCoord and space width) and newline has started (using yCoord) and checks if current font is new font (if it is then sends it to html_class) and finally writes proper syntax to “body.txt” .*
  - *Font processing is also done in this in order to extract font family, font weight, font style, font size and space width and then store this font in new styling class.*

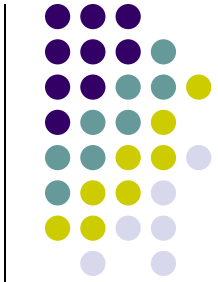
Demo
HTML Parser



- imageParser Class
  - *It parses all the image extracted from the pdf which is stored in “imageLocations.txt” and writes proper syntax to the “body.txt” file which species which image has to be included and its page no. .*

Demo
HTML Class Design





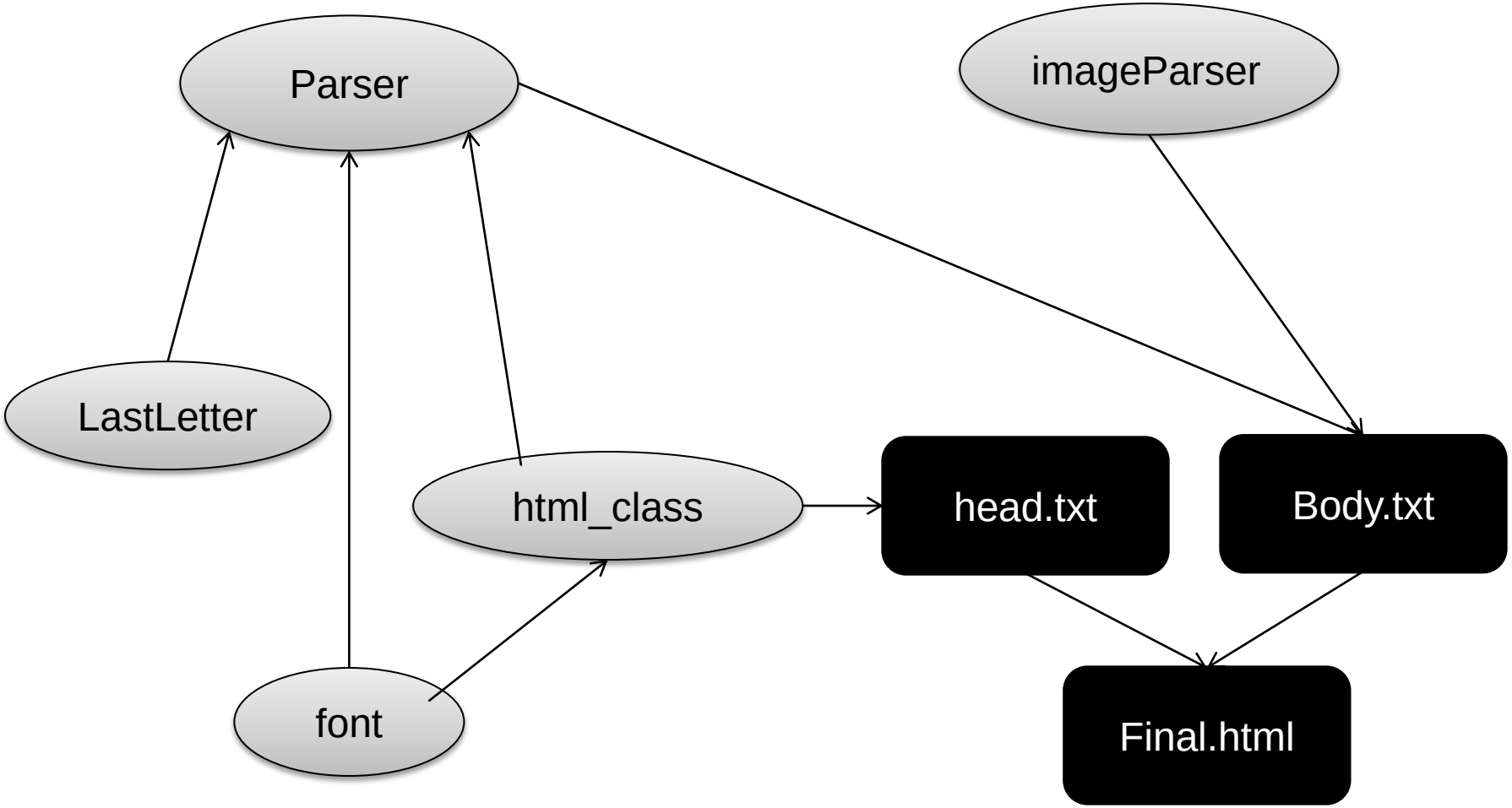
```
595.276 841.89
H 119.821045 97.936 10.9091 8.181824 3.0305479 PNZAKP+CMR10
o 119.821045 106.11782 10.9091 5.4545517 3.0305479 PNZAKP+CMR10
p 119.821045 111.57237 10.9091 6.061096 3.0305479 PNZAKP+CMR10
e 119.821045 117.93893 10.9091 4.848007 3.0305479 PNZAKP+CMR10
t 119.821045 126.41966 10.9091 4.2425537 3.0305479 PNZAKP+CMR10
h 119.821045 130.66222 10.9091 6.061096 3.0305479 PNZAKP+CMR10
i 119.821045 136.72331 10.9091 3.030548 3.0305479 PNZAKP+CMR10
s 119.821045 139.75386 10.9091 4.3025513 3.0305479 PNZAKP+CMR10
l 119.821045 147.68915 10.9091 3.030548 3.0305479 PNZAKP+CMR10
e 119.821045 150.7197 10.9091 4.848007 3.0305479 PNZAKP+CMR10
t 119.821045 155.5677 10.9091 4.2425537 3.0305479 PNZAKP+CMR10
t 119.821045 159.81026 10.9091 4.2425537 3.0305479 PNZAKP+CMR10
e 119.821045 164.05281 10.9091 4.848007 3.0305479 PNZAKP+CMR10
```

textLocations.txt

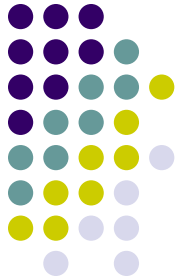
```
0 12.52 15.36 50 50
1 123.25 126.32 50 100
```

imageLocations.txt

Demo
HTML Class Design

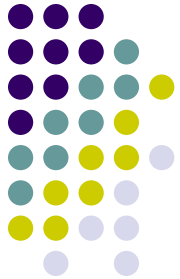


Demo
Latex Parser



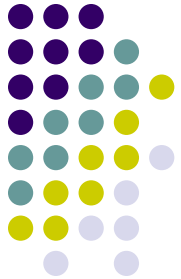
- ParserHandler Class
  - *It generates final latex code*
  - *It contains all the syntax required to show a give type of string format in latex code like “\begin{document} ,using “\item” for bullets, using “\section” for new section and many thing else.*
  - *It also contains unordered map for latex representation of special characters, math symbols and greek letters.*

Demo
Latex Parser



- Parser Class
  - *This is the class which directly extracts a letter and its attributes(like x coord , y coord , font size , font etc) form the pdf Data that was extracted using the java code and pdfBox library.*
  - *Now depending upon the location of this letter it calls the correct method from parser handler object.*
  - *A few cases are eleborated below:*
  - *Let the coordinates of this currently extracted letter be x,y and that of last extracted letter be x1 , y1*

Demo
Latex Parser



- Parser Class
- *if(y = y1){*

  both the letter are in the same line

  if they are in the same word{

  if the fonts are same{add them to the text of same div }

  else {terminate the last div and start a new div}

  }

  else if they are seperated by one space gap{

  if the fonts are same {add this letter to the same div text after adding a space character}

  else {terminate the last div and start a new div}

  }

  Else{terminate the last div and create a new div}

}

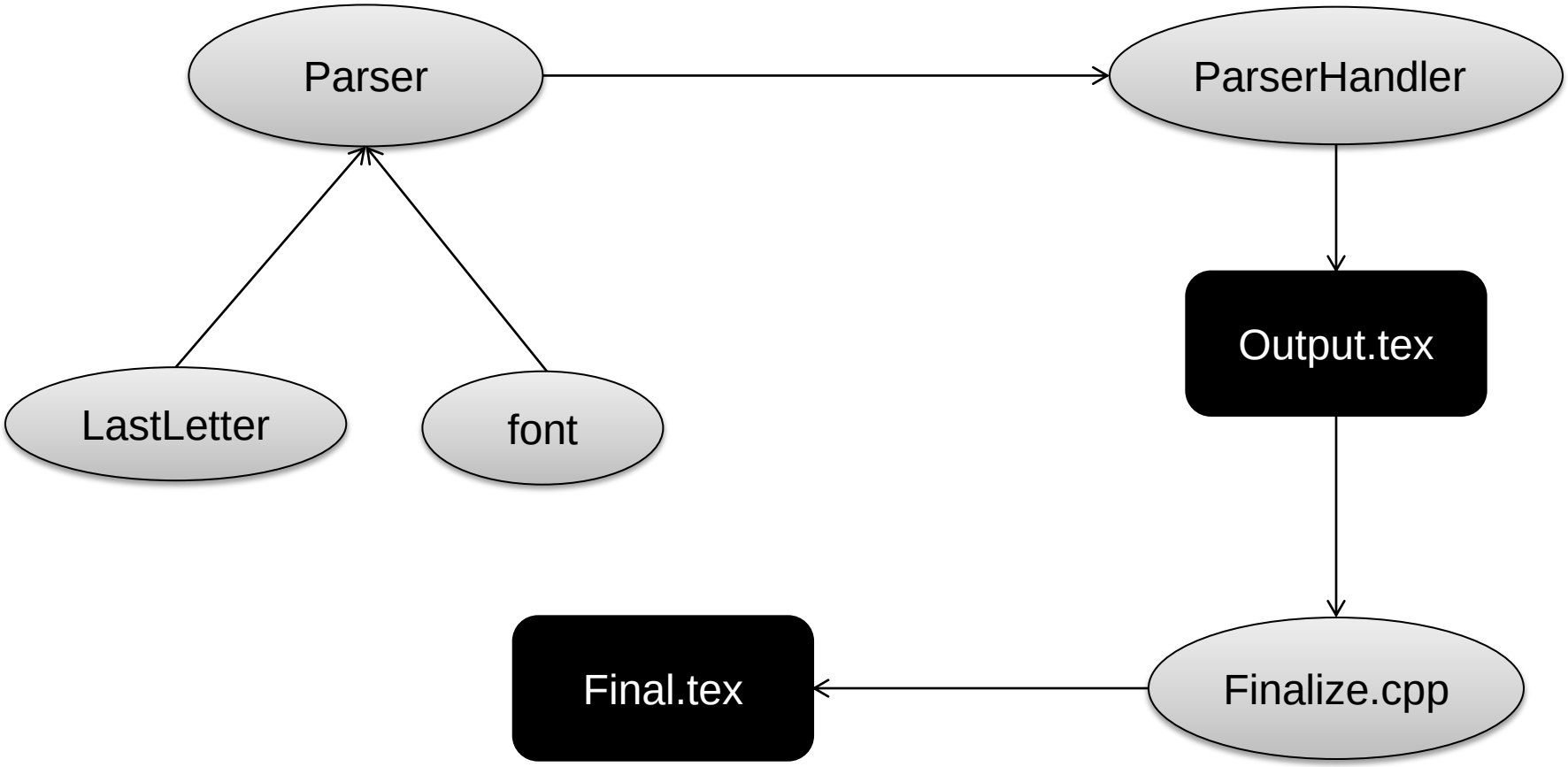
Else{

the two letter are in different line.

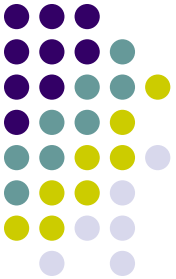
terminate the last div and create a new div

}

Demo
Latex Class Design



Teamwork Details



- Shubham Mehta:

  HTML - imageParser.h, parser.cpp, lastLetter.h

  Latex – parserHandler.h

  Bash – run.sh, compile.sh

- Rahul Singhal

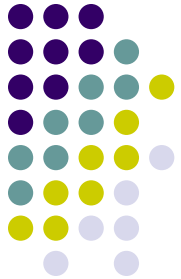
  HTML - font.h, html_class.h

  Latex – parser.cpp

  Java – PrintTextLocation.java, PrintImageLocation.java

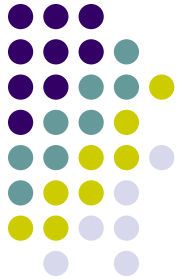
Overall Contribution by Team Member 1	Overall Contribution by Team Member 2
45%	55%

Design Details



- Algorithm
  - *We have made our own parser in which we have identified font, alignment of text, type and level of bullets, starting of section, image position and text position to make it look like a table in case we have detected a table.*
  - *After detecting all these we have inserted required syntax in order to represent text in html and latex format.*
- Implementation
  - Libraries
    - *Pdf-box apache – used for extraction of text and image position in given pdf with corresponding (char) text and image.*
    - *Java default libraries*

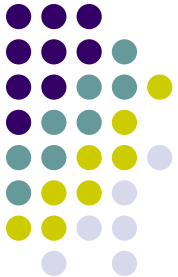
Algorithm Design



Problem: Identify the alignment of the text

- *We stored the starting x coordinate of the line min a variable named lastXCoordinate. Now when the end of this line was reached we found the mid point of this text line as textMidPt = (lastXCoordinate + endXCoordinate)/2. Now we compared it with the mid point of the page.*
- *In case this text was inside some section or bullet object. This implies the entire text was shifted by some pixels to the right. So in this case we found out the mid point of page as pageMidPt = (ActualMidPoint of the page + offset from right due to bullet)*
- *Now to get alignment:*
- *If(textMidPt < PageMidPt – tolerance) left alignment*
- *If(textMidPt >= PageMidPt – tolerance && textMidPt <= PageMidPt + tolerance) center alignment*
- *Else right alignment*
- *This algorithm works for almost all of the very well designed pdf files (created using TEX) . However it sometimes gets confused between center and left or center and right alignment due to some delta(very very small , but it affects the output) tolerance problem.*
- *But since it generates the code without any error , the user may go through the code and verify the line alignments according to him and modify according to his will.*

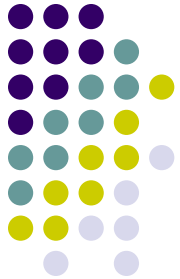
Algorithm Design



Problem: Identify the bulletting pattern of the pdf file just by looking at the positions of the letter extracted from the pdf files.

- *We used stacks to overcome this problem. Whenever bullet was encountered i.e one of A. , (a) , 1. , spherical bullet or i. , its starting X coordinate was pushed in a stack calles bulletStack.*
- *Now when we encountered next bullet , we compared the start x coordinate of this next line with the x coordinate of the bullet at the top of the bulletStack. This creates three cases:*
- *Case 1: (x > bulletStack.top()) this new line is shifted to the right as compared to the last bullet created. This implies a new bullet object has started. Push this new x coordinate in bulletStack*
- *Case 2: (x == bulletStack.top()) This new line is aligned with the last bullet pushed in the stack. This implies it is the next bullet of the last created bullet object.*
- *Case 3: (x < bulletStack.top()) This new line is shifted to the left of the last formed bullet . This implies the end of the last formed bullet. End the lastly formed bullet and pop() the x coordinate of this bullet from the bulletStack. And recursively apply the same algorithm again*

Algorithm Design

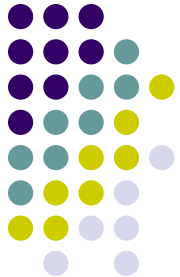


Problem: Identify the special MULTICHARACTER Symbols in the pdf.

- *Symbols like greek variables (α , β , ψ , π , Σ, and many more) are multicharacter in nature. All these symbols are of two of three characters long. In addition some further digging revealed that each of the character of these symbols have negative ASCII value.*
- *We made use of this information to read these special symbols from the extracted pdf data. Whenever we encountered a negative ASCII valued character , we kept on reading characters until we got a positive ASCII valued character.*
- *We stored this symbol in string. Now we could not put these symbols directly in the latex code syntax.*
- *To solve this problem we used unordered maps. We mapped about 30-35 very commonly used greek characters to their latex syntax as shown below:*

  {"β", "$\\beta$"},{"θ", "$\\theta$"},{"ξ","$\\xi$"},{"υ", "$\\upsilon$"}……. and so on.

Algorithm Design



Problem: Generally each page of Pdf files contains a lot of data.We could not extract data for all the pages in one go and store in a single text file. When we tried it once for a file of about 50 pages , the machine froze and we had to reeboot the system.

- *To solve this problem we wrote a bash script to extract and process the pdf file pages one by one.*
- *For example to convert a pdf from page number i to j to html , we first processed ith page of the pdf file and stored the output in a text file. Next we deleted the intermediate files created and ran the same code on page (i+1) and appended the processed output to the last formed output file in a new div tag.*
- *Finally when all the pages got processed we appended the html basic top and bottom syntax(meta data , <html> <style> and other tags) and created the final output file.*
- *Similarly we processed the pdf files and created the corresponding latex syntax , but this time we added \newpage tag whenever we appended the code for next page.*
- *Finally we added the formal package inclusion code and begin{document} and end{documenmt} code to the starting and end of the file respectively.*

Class Design - Details

Class Name	Brief Description
Lastletter	*Stores information about the character (xCoord, yCoord, letterWidth, character …etc)*
Font	*Stores information about the font (family, weight, style, space width..etc)*
Html_class	*Stores information about styling class(font, id…etc)*

Class Design - Details

Class Name	Brief Description
Parser(html)	*Parses textLocations.txt and writes the final string with proper syntax in body.txt file corresponding to text*
ImageParser (html)	*Parses imageLocations.txt and writes the final string with proper syntax in body.txt file corresponding to image*
Parser	*Stores information about styling class(font, id…etc)*

Class Design - Details

Class Name	Brief Description
Parser(latex)	*Parses textLocations.txt and detects the starting of document, bullets, sections, different levels of bullets, text alignment, end of document and calls funcs of parserHandler.*
ParserHandler (latex)	*Again processes the string and adds required syntax for proper latex format and writes string to final.tex*

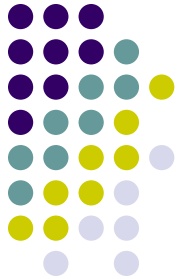
Data Structures Used (Latex)

Purpose for which data structure is used (parser.h)	Data Structure Used	Whether Own Implementation or STL
To start Bullet start Xcoord in order to find the alignment of the text and check if a bullet object has ended making use of indentation of text in pdf.	*Stacks*	*STL*
To store bullet type(itemize, arabic, roman, alphabetic(both cases)) in order to keep track of initialization and termination of bullet object	Stacks	*STL*

Data Structures Used (Latex)

Purpose for which data structure is used (parserHandler.h)	Data Structure Used	Whether Own Implementation or STL
To store latex representation of special characters, greek letters, math letters	*Unordered Map*	*STL*
To store beginning and ending syntax of bullets that is to be wrote in final.tex	Stacks	*STL*

Source Code Information



File Name	Brief Description	Author (Team Member)
Font.h	*Declares Font class and methods*	*Rahul*
LastLetter.h	*Declares LastLetter class and methods*	*Shubham*
Html_class.h	Declares html_class class and methods and writes styling classes	Rahul
imageParser.h	Implements image parsing.	Shubham

Source Code Information

File Name	Brief Description	Author (Team Member)
Parser.cpp(html)	*Implements text parsing*	*Shubham and Rahul*
Combine.cpp	*Combines head.txt and body.txt*	*Shubham*
Parser.cpp (latex)	Implements text parsing and calls funcn of parser handler	Rahul
ParserHandler.cpp (latex)	Implements image parsing.	Shubham

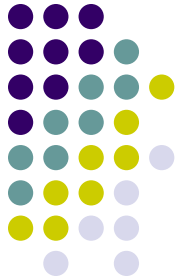
Source Code Information

File Name	Brief Description	Author (Team Member)
Main1.cpp(html)	*Helps in looping through pages*	*Rahul*
Main.cpp (latex)	*Helps in looping through pages*	*Shubham*
Finalize.cpp	Final parsing for minute syntax errors and finalize final.tex file	Shubham
ExtractImageaAndText.java	Main file for text and image extraction	Rahul

Source Code Information

File Name	Brief Description	Author (Team Member)
PrintTextLocation.java	*Extracts text information*	*Rahul*
PrintImageLocation.java	*Extracts image information*	*Rahul*
Compile.sh	Compiles java files	Shubham
Run.sh	Finally executes	Shubham

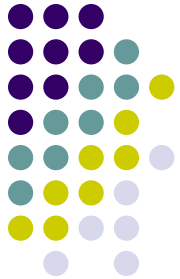
Brief Conclusion



pdfToHTML:

- *The parsing technique and algorithms we used seem to work on almost all the pdf files.*
- *It produces an almost similar looking html file including the tables , bullets , images and text.*
- *However in case of tables the output doesn't have any border around the table but it is perfectly aligned.*
- *The problem in identifying tables is there are no tags that could help figuring out if the next object is table**. Pdf file is actually a graphical representation of the data it contains , doesn't matter whether it is a text or a table or an image. This property of pdf files make them viewable and runnable on any machine independent of the softwares and specifications . This is why pdf files have compression property as it looses information about most of the objects it contains. The same property makes the data extraction very difficult.***
- *in some cases the library we used i.e PdfBox doesn't work well. It extracts all the data from the pdf file but outputs a constant font of size = 1 independent of the actual character/font size in the pdf. In this case the html file created is exactly the same as the input pdf file except the font of each letter is set to 1 pixel and this makes the html file look like an empty file.*
- *We tried to find the bug in the library we have used but could not find anything that helps. However if the library works fine , we guarantee a perfect output.*

Brief Conclusion



pdfToLatex:

- *Almost all the pdf files that were originally generated using latex give a perfect output.*
- *We have included multi-level bullets and enumeration , sections , alignments , and if the file was originally generated using latex , font weight and style also.*
- *We have included about 30-35 special multicharacter (greek characters like alpha , beta , gamma , sigma and other mathematical operators) . Other symbols can be added to the map contained in parserHandler.h file as per requirement.*
- *The latex code that is generated has sometimes an alignment for some of the lines different from that in actual pdf file. The user is requested to modify the generated latex code according to his need.*
- *This ambiguity has crept into our code due to some tolerence problem. Different Users generate pdf files for the same data. Many users add some useless spaces at some places. Sometimes the text get shifted by a few spaces unknowingly. All these mistakes create problems for our algorithm.*
- *To overcome such problems we have given tolerence to all the comparisions.*
- *Almost all the things work fine. Even a pdf file of 50 pages can be converted to latex code and by making a few changes in the latex code and perfect latex file cam be generated.*

Thank You

Hope you enjoyed our presentation

☺