SUBJECT: SYSTEM SOFTWARE
QUESTION BANK

Q.1	What is a Language Processor?
Q.2	Explain Language Processing activities in brief.
Q.3	Explain Life Cycle of a Source Program.
Q.4	Write a short note on Text Editors and its types.
Q.5	What are phases and passes? Compare and contrast.
Q.6	Draw diagram and explain the analysis phase of a Language Processor.
Q.7	Briefly explain IR with an example.
Q.8	Draw diagram and explain the synthesis phase of a Language Processor.
Q.9	Write a Short note on the Symbol Table.
Q.10	What are the different data structures used for symbol table? List them and explain it briefly.
Q.11	Draw a diagram and explain the working of a toy compiler.
Q.12	Write a short note on Program execution activities.
Q.13	What is the role of the program generator domain?
Q.14	What are the different operations provided by the symbol table?
Q.15	Explain Sequential search organization using an example.
Q.16	What are self organizing lists?
Q.17	How is a token searched in search trees? Explain with an example.
Q.18	How does hashing work?
Q.19	What are the allocation data structures used for?
Q.20	Compare and contrast Stack and Heap.
Q.21	What are the characteristics of a system software?

Q.22	What is assembler and assembly language? Give its advantages.
Q.23	What are the basic functionalities provided by an Assembler.
Q.24	What are the types of Assembler?
Q.25	Explain the elements of Assembly code and types of statements.
Q.26	Explain the design of a Two pass assembler
Q.27	How does Single pass assembler work?
Q.28	Define Forward reference and it can be solved.
Q.29	Briefly discuss backpatching.
Q.30	What are the different tables used in Two pass assembler.
Q.31	What are the different ways to write intermediate code?
Q.32	Explain Assembler directives.
Q.33	Write difference between Single pass and Two pass assembler.
Q.34	Define the terms "macro definition" and "macro call."
Q.35	What is macro expansion? Explain with a simple example.
Q.36	Describe the purpose of a macro processor.
Q.37	What are the basic tasks performed by a macro processor?
Q.38	Explain the process of macro expansion with a step-by-step example.
Q.39	What are nested macro calls? Provide an example.
Q.40	Describe the design of a simple macro preprocessor. What functions does it perform?
Q.41	How do advanced macro facilities enhance the basic macro processing capabilities?
Q.42	Discuss the functions of a macro processor and their importance in assembler design.
Q.43	Describe the design issues involved in creating a macro processor.

Q.44	How do these issues affect performance and flexibility?
Q.45	Discuss the architecture of a macro assembler. How does it differ from a traditional assembler?
Q.46	What is a compiler? Provide a brief definition.
Q.47	List the main phases of a compiler.
Q.48	Define the term "lexical analysis."
Q.49	What is the primary role of a syntax analyzer?
Q.50	Name three types of compilers and give a brief description of each.
Q.51	Explain the differences between a compiler, interpreter, and assembler with examples. What are the advantages and disadvantages of each?
Q.52	Describe the purpose of semantic analysis in a compiler. How does it differ from syntax analysis?
Q.53	What is the difference between top-down and bottom-up parsing?
Q.54	Describe the function of a symbol table in a compiler.
Q.55	What are the benefits of code optimization in compilers?
Q.56	Discuss the different phases of a compiler in detail. What transformations occur at each phase? Illustrate with examples.
Q.57	Explain the concept of a parse tree and its role in parsing. Provide an example of how a parse tree is constructed from a given grammar.
Q.58	What is left recursion, and why is it problematic in parsing?
Q.59	Describe two code optimization techniques and their impact on execution speed.
Q.60	What is the role of intermediate code generation in a compiler?
Q.61	Describe the various parsing techniques (LL, LR, SLR, recursive descent) and their suitability for different types of grammars. Include examples of when to use each technique.
Q.62	What is an LL parser? How does it operate?

Q.63	Define left factoring and its significance in parsing.
Q.64	Define and explain operator precedence parsing.
Q.65	What is interpretation in the context of programming languages?
Q.66	List two benefits of using an interpreted language.
Q.67	Define the Java Language Environment.
Q.68	What is the Java Virtual Machine (JVM)?
Q.69	Name three types of errors that can occur in Java programs.
Q.70	Explain the concept of interpretation versus compilation. What are the key differences?
Q.71	Describe the role of the Java Virtual Machine in executing Java programs.
Q.72	What are the primary responsibilities of the JVM?
Q.73	Explain what runtime errors are, and provide an example.
Q.74	What is a syntax error? How does it differ from a semantic error?
Q.75	Discuss the various types of errors in Java (syntax, runtime, and logical). Provide examples of each and explain how they can be identified.
Q.76	Describe the debugging procedures typically followed when debugging a Java application. What tools and techniques can be used?
Q.77	What is a dynamic debugger? How does it differ from a static debugger?
Q.78	Explain the classification of debuggers and their functions.
Q.79	Analyze the features and benefits of using an interactive debugger in Java development. How does it enhance the debugging process?
Q.80	Discuss the significance of debugging in software development. What are some common debugging strategies and best practices?
Q.81	Explain the process of interpretation in detail. How does an interpreter execute code differently compared to a compiler? Illustrate with an example where interpretation is more beneficial than compilation.
Q.82	Describe the architecture and functioning of the Java Virtual Machine

	(JVM). Include a discussion on how the JVM handles memory management, garbage collection, and execution of Java bytecode. Provide a diagram that explains the internal components of the JVM.
Q.83	Discuss in detail the different types of errors that can occur in Java programs, including syntax errors, runtime errors, and logical errors. Provide examples of each and explain the best practices for identifying and resolving these errors in a development environment.
Q.84	Provide an in-depth explanation of Java's error-handling mechanisms, focusing on exceptions. Discuss the different types of exceptions (checked and unchecked), the try-catch mechanism, and how custom exceptions are implemented. How does proper exception handling contribute to robust software development?
Q.85	Provide a detailed classification of debuggers. Discuss the different types (e.g., static, dynamic, post-mortem debuggers) and explain their respective roles in the development process. How do they assist developers in handling various types of program errors?
Q.86	What is a linker?
Q.87	Define relocation in the context of linking.
Q.88	What is a loader? Explain its primary function.
Q.89	Differentiate between static and dynamic linking.
Q.90	What is a self-relocating program?Explain the basic purpose of a linker in system software. How does it assist in managing memory addresses during program execution?
Q.91	Describe the different types of loading schemes. Provide examples for each. What is the concept of relocation in linking?
Q.92	How does dynamic linking differ from static linking? Provide an example.
Q.93	What is the purpose of a compile-and-go loader?
Q.94	Explain the concept of an overlay structured program.
Q.95	Describe the role of a linker in the Linux operating system. Discuss the design and functionality of self-relocating programs. How do they contribute to efficient memory utilization during program execution?

Q.96	Explain the process of dynamic linking and its advantages over static linking. Include a discussion on how this impacts the size and efficiency of executable programs. What are sequential and direct loaders? How do they differ in their approaches to loading programs into memory?
Q.97	How do modern loaders handle dynamic linking? Give examples from popular operating systems like Linux.
Q.98	What are the challenges in designing a linker for overlay structured programs? Provide a detailed explanation of the relocation process in linking. How do linkers manage external and internal addresses to facilitate program execution?
Q.99	Analyze the design of a modern linker. Discuss the different tasks it performs, such as symbol resolution and relocation. Provide examples from Linux linking processes.
Q.100	Explore the concept of dynamic linking in detail. What are the benefits of dynamic linking, particularly in large-scale software systems? How does it improve memory usage and reduce program size?