

**Question bank Pattern**

**[subject code] [ subject name]**

**Unit 4 [ POINTERS,STRINGS , STRUCTURES & UNIONS ]**

**PART A (10x2)**

Question 1: Explain the declaration and initialization of a pointer in C.

Answer 1: In C, a pointer is declared by specifying its data type, followed by an asterisk (\*), and then the pointer variable's name. Initialization can be done by assigning the address of a variable to the pointer variable. For example:

int \*ptr; // Declaration

int num = 10;

ptr = &num; // Initialization

Question 2: What is a NULL pointer in C, and why is it important?

Answer 2: A NULL pointer in C is a pointer that does not point to any memory location. It is represented as NULL. NULL pointers are important because they are used to indicate that a pointer does not currently refer to a valid memory address, helping to avoid undefined behavior.

Question 3: Explain the concept of pointer arithmetic in C.

Answer 3: Pointer arithmetic in C involves performing arithmetic operations (addition, subtraction) on pointers. When adding or subtracting an integer value to/from a pointer, it moves the pointer to a new memory location based on the size of the data type the pointer is pointing to. For example:

int arr[5];

int \*ptr = arr;

ptr = ptr + 2; // Moves ptr to the 3rd element of arr

Question 4: What is a void pointer in C, and what is its use?

Answer 4: A void pointer (or void\*) in C is a pointer that does not have a specific data type associated with it. It can be used to store the address of any data type. However, to access the data pointed to by a void pointer, it needs to be cast to the appropriate data type.

Question 5: Explain the relationship between arrays and pointers in C.

Answer 5: In C, arrays and pointers are closely related. An array name can be used as a pointer to the first element of the array. For example, int arr[5]; can be treated as a pointer to an integer. Additionally, pointer arithmetic can be applied to arrays to traverse their elements.

Question 6: What is dynamic memory allocation in C, and how is it achieved?

Answer 6: Dynamic memory allocation in C involves allocating memory at runtime using functions like malloc, calloc, or realloc from the stdlib.h library. It allows you to allocate memory as needed and is commonly used for data structures like linked lists and dynamic arrays. You must release dynamically allocated memory using free to avoid memory leaks.

Question 7: Name the four storage classes in C.

Answer 7: The four storage classes in C are: auto, register, static, and extern.

Question 8: What are C strings, and how are they represented?

Answer 8: C strings are arrays of characters terminated by a null character ('\0'). They are used to represent text in C. For example, "Hello" is a C string.

Question 9: Explain the concept of structures in C.

Answer 9: A structure in C is a composite data type that allows you to group multiple variables of different data types under a single name. It is defined using the struct keyword. For example:

struct Person {

char name[50];

int age;

};

Question 10: What are unions in C, and how are they different from structures?

Answer 10: Unions in C are similar to structures, but they share the same memory location for all their members. Only one member of a union can have a value at a time. This makes unions useful when you need to represent different types of data in the same memory location efficiently. Unlike structures, where all members have separate memory locations, a union's size is determined by the largest member.

**PART B (5x13)**

Question 1: Program Dynamic Memory Allocation with Pointers.

Answer 1:

#include <stdio.h>

#include <stdlib.h>

int main() {

int n;

printf("Enter the number of integers to store: ");

scanf("%d", &n);

// Allocate memory dynamically for an array of integers

int \*arr = (int \*)malloc(n \* sizeof(int));

if (arr == NULL) {

printf("Memory allocation failed.");

return 1;

}

// Read integers from the user

printf("Enter %d integers:\n", n);

for (int i = 0; i < n; i++) {

scanf("%d", &arr[i]);

}

// Display the entered integers

printf("Entered integers: ");

for (int i = 0; i < n; i++) {

printf("%d ", arr[i]);

}

// Free the dynamically allocated memory

free(arr);

return 0;

}

Question 2: Write a C program that defines a structure Person to store information about a person, including their name, age, and height. The program should use dynamic memory allocation to create an array of Person structures and then display the details of each person.

Answer 2:

#include <stdio.h>

#include <stdlib.h>

struct Person {

char name[50];

int age;

float height;

};

int main() {

int n;

printf("Enter the number of people: ");

scanf("%d", &n);

struct Person \*people = (struct Person \*)malloc(n \* sizeof(struct Person));

if (people == NULL) {

printf("Memory allocation failed.");

return 1;

}

for (int i = 0; i < n; i++) {

printf("Enter name, age, and height for person %d: ", i + 1);

scanf("%s %d %f", people[i].name, &people[i].age, &people[i].height);

}

printf("Details of people:\n");

for (int i = 0; i < n; i++) {

printf("Person %d: Name: %s, Age: %d, Height: %.2f\n", i + 1, people[i].name, people[i].age, people[i].height);

}

free(people);

return 0;

}

Question 3: Write a C program that takes a user-input string and uses pointers to reverse the string in place. Display the reversed string.

Answer:

#include <stdio.h>

int main() {

char str[100];

printf("Enter a string: ");

scanf("%s", str);

char \*start = str;

char \*end = start;

while (\*end != '\0') {

end++;

}

end--;

while (start < end) {

char temp = \*start;

\*start = \*end;

\*end = temp;

start++;

end--;

}

printf("Reversed string: %s\n", str);

return 0;

}

Question 4: Write a C program that uses a union to convert a floating-point value to its binary representation and vice versa. The program should allow the user to input either a float or a binary representation, perform the conversion, and display the result.

Answer:

#include <stdio.h>

union FloatConverter {

float floatValue;

unsigned int binaryValue;

};

int main() {

union FloatConverter converter;

char choice;

printf("Enter 'f' to input a float, 'b' to input binary: ");

scanf(" %c", &choice);

if (choice == 'f') {

printf("Enter a floating-point value: ");

scanf("%f", &converter.floatValue);

printf("Binary representation: %u\n", converter.binaryValue);

} else if (choice == 'b') {

printf("Enter a binary representation: ");

scanf("%u", &converter.binaryValue);

printf("Floating-point value: %.2f\n", converter.floatValue);

} else {

printf("Invalid choice.\n");

}

return 0;

}

Question 5: Write a C program that demonstrates the use of different storage classes - auto, register, static, and extern. The program should show the scope and behavior of variables with different storage classes.

Answer:

#include <stdio.h>

int globalVar = 10;

void function() {

auto int localVarAuto = 20;

printf("auto variable in function: %d\n", localVarAuto);

static int localVarStatic = 30;

printf("static variable in function: %d\n", localVarStatic);

}

int main() {

register int localVarRegister = 40;

printf("register variable in main: %d\n", localVarRegister);

extern int globalVar;

printf("extern variable in main: %d\n", globalVar);

function();

return 0;

}