**Q1. Write a C program to implement priority queue using doubly linked**

**list (Priority depends on identitynumber. Small identity number has greater priority. If identity numbers are equal. Then FIFO rules are used) with following functions:**

**1) insert 2) serve 3) display**

#include <stdio.h>

#include <stdlib.h>

typedef struct Node {

int id;

struct Node\* prev;

struct Node\* next;

} Node;

Node\* insert(Node\* head, int id) {

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->id = id;

newNode->next = newNode->prev = NULL;

if (!head) {

return newNode;

}

Node\* temp = head;

Node\* prev = NULL;

while (temp && temp->id <= id) {

prev = temp;

temp = temp->next;

}

if (!prev) {

newNode->next = head;

head->prev = newNode;

return newNode;

}

newNode->next = prev->next;

newNode->prev = prev;

if (prev->next) {

prev->next->prev = newNode;

}

prev->next = newNode;

return head;

}

Node\* serve(Node\* head) {

if (!head) {

printf("Queue is empty.\n");

return NULL;

}

printf("Served: %d\n", head->id);

Node\* temp = head;

head = head->next;

if (head) head->prev = NULL;

free(temp);

return head;

}

void display(Node\* head) {

Node\* temp = head;

printf("Queue: ");

while (temp) {

printf("%d ", temp->id);

temp = temp->next;

}

printf("\n");

}

int main() {

Node\* head = NULL;

int choice, id;

do {

printf("\n1. Insert\n2. Serve\n3. Display\n4. Exit\nEnter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter ID: ");

scanf("%d", &id);

head = insert(head, id);

break;

case 2:

head = serve(head);

break;

case 3:

display(head);

break;

}

} while (choice != 4);

return 0;

}

**OUTPUT:**

1. Insert

2. Serve

3. Display

4. Exit

Enter your choice: 1

Enter ID: 123

1. Insert

2. Serve

3. Display

4. Exit

Enter your choice: 1

Enter ID: 357

1. Insert

2. Serve

3. Display

4. Exit

Enter your choice: 2

Served: 123

1. Insert

2. Serve

3. Display

4. Exit

Enter your choice: 3

Queue: 357

1. Insert

2. Serve

3. Display

4. Exit

Enter your choice: 1

Enter ID: 951

1. Insert

2. Serve

3. Display

4. Exit

Enter your choice: 3

Queue: 357 951

1. Insert

2. Serve

3. Display

4. Exit

**Que2: Write a C program to create a single linked list then input a value V,**

**arrange the linked list, such that all nodes less than V come before nodes greater than or equal to V in the linked list.**

#include <stdio.h>

#include <stdlib.h>

typedef struct Node {

int data;

struct Node\* next;

} Node;

Node\* insertEnd(Node\* head, int data) {

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->data = data;

newNode->next = NULL;

if (!head) return newNode;

Node\* temp = head;

while (temp->next) temp = temp->next;

temp->next = newNode;

return head;

}

void display(Node\* head) {

Node\* temp = head;

while (temp) {

printf("%d -> ", temp->data);

temp = temp->next;

}

printf("NULL\n");

}

Node\* rearrange(Node\* head, int V) {

Node \*lessHead = NULL, \*lessTail = NULL;

Node \*moreHead = NULL, \*moreTail = NULL;

Node \*temp = head;

while (temp) {

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->data = temp->data;

newNode->next = NULL;

if (temp->data < V) {

if (!lessHead) lessHead = lessTail = newNode;

else {

lessTail->next = newNode;

lessTail = newNode;

}

} else {

if (!moreHead) moreHead = moreTail = newNode;

else {

moreTail->next = newNode;

moreTail = newNode;

}

}

temp = temp->next;

}

if (!lessHead) return moreHead;

lessTail->next = moreHead;

return lessHead;

}

int main() {

Node\* head = NULL;

int n, value, V;

printf("Enter number of elements: ");

scanf("%d", &n);

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

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

scanf("%d", &value);

head = insertEnd(head, value);

}

printf("Enter value V: ");

scanf("%d", &V);

printf("Original List:\n");

display(head);

head = rearrange(head, V);

printf("Rearranged List:\n");

display(head);

return 0;

}

**OUTPUT:**

Enter number of elements: 5

Enter 5 elements:

7 2 9 1 5

Enter value V: 5

Original List:

7 -> 2 -> 9 -> 1 -> 5 -> NULL

Rearranged List:

2 -> 1 -> 5 -> 7 -> 9 -> NULL

**Que:** **Write a C program to create two single linked lists, and then write**

**another function to subtract two numbers represented as linked list.**

**List1 -> 5 -> 6 -> 8 -> NULL**

**List2 -> 2 -> 3 -> 4 -> NULL**

**Output -> 334**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

typedef struct Node {

int data;

struct Node\* next;

} Node;

Node\* insertEnd(Node\* head, int data) {

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->data = data;

newNode->next = NULL;

if (!head) return newNode;

Node\* temp = head;

while (temp->next) temp = temp->next;

temp->next = newNode;

return head;

}

void display(Node\* head) {

while (head) {

printf("%d", head->data);

head = head->next;

}

printf("\n");

}

int linkedListToNumber(Node\* head) {

int num = 0;

while (head) {

num = num \* 10 + head->data;

head = head->next;

}

return num;

}

Node\* numberToLinkedList(int num) {

if (num == 0) return insertEnd(NULL, 0);

Node\* head = NULL;

int digits[20], i = 0;

while (num) {

digits[i++] = num % 10;

num /= 10;

}

for (int j = i - 1; j >= 0; j--) {

head = insertEnd(head, digits[j]);

}

return head;

}

int main() {

Node \*list1 = NULL, \*list2 = NULL;

int n1, n2, val;

printf("Enter number of digits in first number: ");

scanf("%d", &n1);

printf("Enter digits of first number:\n");

for (int i = 0; i < n1; i++) {

scanf("%d", &val);

list1 = insertEnd(list1, val);

}

printf("Enter number of digits in second number: ");

scanf("%d", &n2);

printf("Enter digits of second number:\n");

for (int i = 0; i < n2; i++) {

scanf("%d", &val);

list2 = insertEnd(list2, val);

}

int num1 = linkedListToNumber(list1);

int num2 = linkedListToNumber(list2);

if (num1 < num2) {

printf("Error: Negative result not supported.\n");

return 0;

}

int result = num1 - num2;

printf("Output: ");

Node\* resultList = numberToLinkedList(result);

display(resultList);

return 0;

}

**OUTPUT:**

Enter number of digits in first number: 4

Enter digits of first number:

9 8 7 6

Enter number of digits in second number: 3

Enter digits of second number:

5 4 3

Output: 9333

**Que4: Write a C program to create a single linked list , like**

**L0 -> L1 ->... -> Ln-1 -> Ln**

**Write another C function to rearrange the nodes in the list so that the new formed list**

**is:**

**L0 -> Ln -> L1 -> Ln-1 -> L2-> Ln-2.**

#include <stdio.h>

#include <stdlib.h>

typedef struct Node {

int data;

struct Node\* next;

} Node;

Node\* insertEnd(Node\* head, int data) {

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->data = data;

newNode->next = NULL;

if (!head) return newNode;

Node\* temp = head;

while (temp->next)

temp = temp->next;

temp->next = newNode;

return head;

}

void display(Node\* head) {

while (head) {

printf("%d -> ", head->data);

head = head->next;

}

printf("NULL\n");

}

Node\* reverse(Node\* head) {

Node \*prev = NULL, \*curr = head, \*next;

while (curr) {

next = curr->next;

curr->next = prev;

prev = curr;

curr = next;

}

return prev;

}

Node\* mergeAlternate(Node\* first, Node\* second) {

Node \*head = first, \*temp1, \*temp2;

while (first && second) {

temp1 = first->next;

temp2 = second->next;

first->next = second;

if (!temp1) break;

second->next = temp1;

first = temp1;

second = temp2;

}

return head;

}

Node\* rearrange(Node\* head) {

if (!head || !head->next) return head;

Node \*slow = head, \*fast = head;

while (fast->next && fast->next->next) {

slow = slow->next;

fast = fast->next->next;

}

Node\* second = slow->next;

slow->next = NULL;

second = reverse(second);

return mergeAlternate(head, second);

}

int main() {

Node\* head = NULL;

int n, val;

printf("Enter number of elements: ");

scanf("%d", &n);

printf("Enter elements:\n");

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

scanf("%d", &val);

head = insertEnd(head, val);

printf("Original List:\n");

display(head);

head = rearrange(head);

printf("Rearranged List:\n");

display(head);

return 0;

}

**OUTPUT:**

Enter number of elements: 6

Enter elements:

9 8 7 6 5 4

Original List:

9 -> 8 -> 7 -> 6 -> 5 -> 4 -> NULL

Rearranged List:

9 -> 4 -> 8 -> 5 -> 7 -> 6 -> NULL

**Que5:** **Write a C program to add two quadratic equations, using linked**

**list.**

#include <stdio.h>

#include <stdlib.h>

typedef struct Node {

int coeff;

int power;

struct Node\* next;

} Node;

Node\* insertTerm(Node\* head, int coeff, int power) {

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->coeff = coeff;

newNode->power = power;

newNode->next = NULL;

if (!head || power > head->power) {

newNode->next = head;

return newNode;

}

Node\* temp = head;

while (temp->next && temp->next->power >= power)

temp = temp->next;

if (temp->power == power) {

temp->coeff += coeff;

free(newNode);

} else {

newNode->next = temp->next;

temp->next = newNode;

}

return head;

}

void display(Node\* head) {

Node\* temp = head;

while (temp) {

printf("%dx^%d", temp->coeff, temp->power);

if (temp->next) printf(" + ");

temp = temp->next;

}

printf("\n");

}

Node\* addPolynomials(Node\* poly1, Node\* poly2) {

Node\* result = NULL;

while (poly1) {

result = insertTerm(result, poly1->coeff, poly1->power);

poly1 = poly1->next;

}

while (poly2) {

result = insertTerm(result, poly2->coeff, poly2->power);

poly2 = poly2->next;

}

return result;

}

int main() {

Node \*poly1 = NULL, \*poly2 = NULL;

int n, coeff, power;

printf("Enter number of terms in first polynomial: ");

scanf("%d", &n);

printf("Enter each term as: coefficient power\n");

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

scanf("%d%d", &coeff, &power);

poly1 = insertTerm(poly1, coeff, power);

}

printf("Enter number of terms in second polynomial: ");

scanf("%d", &n);

printf("Enter each term as: coefficient power\n");

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

scanf("%d%d", &coeff, &power);

poly2 = insertTerm(poly2, coeff, power);

}

printf("\nPolynomial 1: ");

display(poly1);

printf("Polynomial 2: ");

display(poly2);

Node\* sum = addPolynomials(poly1, poly2);

printf("Sum of Polynomials: ");

display(sum);

return 0;

}

**OUTPUT:**

Enter number of terms in first polynomial: 4

Enter each term as: coefficient power

2 4 3 3 4 1 1 0

Enter number of terms in second polynomial: 3

Enter each term as: coefficient power

5 4 2 2 7 0

Polynomial 1: 2x^4 + 3x^3 + 4x^1 + 1x^0

Polynomial 2: 5x^4 + 2x^2 + 7x^0

Sum of Polynomials: 7x^4 + 3x^3 + 2x^2 + 4x^1 + 8x^0

**Que6:** **Using circular Queue allocate time slots of 10ns for given processes**

**in time sharing environment and then print which process will be completed in how much time.**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#define TIME\_SLOT 10

#define MAX 100

typedef struct {

    int id;

    int burst\_time;

    int remaining\_time;

    int completed\_time;

    bool completed;

} Process;

int main() {

    Process processes[MAX];

    int n, i;

    int total\_time = 0, completed = 0;

    printf("Enter number of processes: ");

    scanf("%d", &n);

    for (i = 0; i < n; i++) {

        printf("Enter burst time for process P%d: ", i + 1);

        scanf("%d", &processes[i].burst\_time);

        processes[i].remaining\_time = processes[i].burst\_time;

        processes[i].id = i + 1;

        processes[i].completed = false;

        processes[i].completed\_time = 0;

    }

    int time = 0;

    while (completed < n) {

        for (i = 0; i < n; i++) {

            if (!processes[i].completed && processes[i].remaining\_time > 0) {

                if (processes[i].remaining\_time <= TIME\_SLOT) {

                    time += processes[i].remaining\_time;

                    processes[i].remaining\_time = 0;

                    processes[i].completed = true;

                    processes[i].completed\_time = time;

                    completed++;

                } else {

                    time += TIME\_SLOT;

                    processes[i].remaining\_time -= TIME\_SLOT;

                }

            }

        }

    }

    printf("\nProcess Completion Times:\n");

    for (i = 0; i < n; i++) {

        printf("Process P%d: %d ns\n", processes[i].id, processes[i].completed\_time);

    }

    return 0;

}

**OUTPUT:**

Enter number of processes: 6

Enter burst time for process P1: 4

Enter burst time for process P2: 10

Enter burst time for process P3: 3

Enter burst time for process P4: 5

Enter burst time for process P5: 8

Enter burst time for process P6: 6

Process Completion Times:

Process P1: 4 ns

Process P2: 14 ns

Process P3: 17 ns

Process P4: 22 ns

Process P5: 30 ns

Process P6: 36 ns

Q1.Describe the various ways to read a string in ‘C’ language with their suitable syntax.  
Write a ‘C’ program to read an alphanumeric string. Display the string by converting digits into words.

#include <stdio.h>

#include <string.h>

void printDigitAsWord(char ch) {

switch(ch) {

case '0': printf("zero "); break;

case '1': printf("one "); break;

case '2': printf("two "); break;

case '3': printf("three "); break;

case '4': printf("four "); break;

case '5': printf("five "); break;

case '6': printf("six "); break;

case '7': printf("seven "); break;

case '8': printf("eight "); break;

case '9': printf("nine "); break;

default: putchar(ch); break;}}

int main() {

char str[200];

printf("Enter a string: ");

fgets(str, sizeof(str), stdin);

printf("Output String:\n");

for (int i = 0; i < strlen(str); i++) {

if (str[i] >= '0' && str[i] <= '9') {

printDigitAsWord(str[i]);

} else {

putchar(str[i]);} }

return 0;

}

**Sample Input:**

Pincode of my city is 248001

Sample Output:

Pincode of my city is two four eight zero zero one

Q1(b) Write a code in ‘C’ to read a multiword string. Input two indexes a and b. Reverse the string starting from index a to index b. Print the final string.

#include <stdio.h>

#include <string.h>

void reverseSubstring(char str[], int a, int b) {

char temp;

while (a < b) {

temp = str[a];

str[a] = str[b];

str[b] = temp;

a++;

b--;}}

int main() {

char str[200];

int a, b;

printf("Enter the string: ");

fgets(str, sizeof(str), stdin);

printf("Enter index a: ");

scanf("%d", &a);

printf("Enter index b: ");

scanf("%d", &b);

reverseSubstring(str, a, b);

printf("Output String:\n%s", str);

return 0;

}

**Sample Input:**

Input String: Delhi is the capital of India

Input index(a): 14

Input index(b): 20

Sample Output:

Delhi is the latipac of India

Q2 Find the Output with an explanation.

**#include <stdio.h>**

**#include <string.h>**

**int main() {**

**char str[] = "how are you?";**

**int i, j, t, len;**

**len = strlen(str);**

**for(i = 0, j = len - 1; i < j; i++, j--) {**

**t = str[i];**

**str[i] = str[j];**

**str[j] = t;**

**}**

**printf("%s", str);**

**return 0;**

**}**

Output(a):

?uoy era woh

**Explanation:**

This code reverses the string "how are you?" by swapping characters from the beginning and end using a loop until the midpoint is reached.

Q2 Find the Output with an explanation.

#include <stdio.h>

int main() {

int arr[] = {0, 1, 2, 3, 4, 5, 10};

static int x, y, k;

int \*p = arr;

for(k = 0; k < 7; k++) {

if (\*(p + k) % 2 == 0 && \*(p + k) % 5 == 0)

++x;

else

++y;

}

printf("%d %d %d", x, y, x - y);

return 0;

}

Output(b):

2 5 -3

**Explanation:**

* Array = {0, 1, 2, 3, 4, 5, 10}
* Elements divisible by both 2 and 5: 0 and 10 → x = 2
* Other elements → y = 5
* Final output = x, y, x - y = 2 5 -3

Q3(a). Draw a flowchart to read a multiword string. Copy this string to another string without spaces. Display the final copied string.

#include <stdio.h>

int main() {

char str1[100] = "my name is Saiyam tuteja";

char str2[100];

int i, j = 0;

for(i = 0; str1[i] != '\0'; i++) {

if(str1[i] != ' ') {

str2[j++] = str1[i];

}

}

str2[j] = '\0';

printf("Output String: %s\n", str2);

return 0;

}

Output :

mynameissaiyamtuteja

Q3(b). Draw a flowchart to input elements into a matrix of size m×n.  
Print the maximum element of each column of matrix.

#include <stdio.h>

int main() {

int rows = 3, cols = 4;

int mat[3][4] = {

{34, 56, 12, 89},

{23, 45, 78, 38},

{15, 21, 67, 92}

};

for(int j = 0; j < cols; j++) {

int max = mat[0][j];

for(int i = 1; i < rows; i++) {

if(mat[i][j] > max) {

max = mat[i][j];

}

}

printf("Maximum of column %d: %d\n", j + 1, max);

}

return 0;

}

Output:

Maximum of column 1: 34

Maximum of column 2: 56

Maximum of column 3: 78

Maximum of column 4: 92

Q4(a). To solve a mathematical problem, Aman inputted elements into a matrix of size m×n.  
Write a ‘C’ program to help him to check whether he created a Sparse matrix or not.  
**Note:** Sparse matrix is a matrix where most of the elements are zero.

#include <stdio.h>

int main() {

int mat[4][3] = {

{1, 0, 0},

{2, 0, 0},

{0, 0, 5},

{0, 3, 0}

};

int rows = 4, cols = 3;

int zeroCount = 0, total = rows \* cols;

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

for(int j = 0; j < cols; j++) {

if(mat[i][j] == 0) {

zeroCount++;

}}}

if(zeroCount > total / 2) {

printf("Sparse Matrix\n");

} else {

printf("Not a Sparse Matrix\n");

}

return 0;

}

Output:

Sparse Matrix

Q4(b). Write a ‘C’ program to input elements into a matrix of size m×n.  
Input two row indices and interchange elements of both rows.

#include <stdio.h>

int main() {

int rows = 5, cols = 3;

int mat[5][3] = {

{2, 5, 7},

{4, 6, 8},

{1, 7, 9},

{3, 5, 2},

{8, 9, 3}

};

int r1 = 1, r2 = 3; // 0-based indexing

for(int j = 0; j < cols; j++) {

int temp = mat[r1][j];

mat[r1][j] = mat[r2][j];

mat[r2][j] = temp;}

printf("Final matrix after interchange:\n");

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

for(int j = 0; j < cols; j++) {

printf("%d ", mat[i][j]);

} printf("\n");

}

return 0;

}

**Output:**

Final matrix after interchange:

2 5 7

3 5 2

1 7 9

4 6 8

8 9 3

Q5(a). Describe various types of pointers available in C programming language.  
How is **call by reference** different from **call by value**?  
Explain with an example.

**Types of Pointers in C:**

1. **Null Pointer** – Initialized to NULL.
2. **Void Pointer** – Can store address of any data type.
3. **Wild Pointer** – Uninitialized pointer.
4. **Dangling Pointer** – Points to memory which has been freed.
5. **Function Pointer** – Points to a function.

Call by value :

#include <stdio.h>

void add(int x) {

x += 10;

printf("Inside function: %d\n", x);

}

int main() {

int a = 5;

add(a);

printf("Outside function: %d\n", a);

return 0;

}

Output:

Inside function: 15

Outside function: 5

Call by reference:

#include <stdio.h>

void add(int \*x) {

\*x += 10;

}

int main() {

int a = 5;

add(&a);

printf("After function call: %d\n", a);

return 0;

}

Output:

After function call: 15

Q5(b). Write a C program to read two 1D arrays.  
Create a new array by finding the intersection of the two input arrays.  
**Access elements using pointer.**

#include <stdio.h>

int main() {

int arr1[] = {2, 4, 6, 8};

int arr2[] = {3, 4, 6, 8, 9, 5};

int arr3[10];

int n1 = 4, n2 = 6, k = 0;

for(int i = 0; i < n1; i++) {

for(int j = 0; j < n2; j++) {

if(\*(arr1 + i) == \*(arr2 + j)) {

arr3[k++] = \*(arr1 + i);

break;

}

}

}

printf("Elements of output array(arr3): ");

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

printf("%d ", arr3[i]);

}

return 0;

}

Input :

Number of elements in array1: 4

Elements of array1: 2 4 6 8

Number of elements in array2: 6

Elements of array2: 3 4 6 8 9 5

Output:

Elements of output array (array3): 4 6 8