**Dynamic memory allocation**

**Programs**

1. **WAP to read and display elements of 1D array using Dynamic memory allocation (using malloc())**

#include<stdio.h>

#include<stdlib.h>

int main( )

{

int \*p,n,i;

printf ("Enter the number of integers to be entered ") ;

scanf("%d" ,&n) ;

p=(int\*)malloc(n\*sizeof(int));//malloc() returns void\* so we need to typecast with the specific data type

if(p==NULL)

{

printf("Memory not available\n");

exit(1);

}

else

{

printf("\n Mmeory allocation was successful");

printf ("\nEnter integer values ") ;

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

{

scanf("%d",p+i);//In place of p+i we can write &p[i](treating it as ID array)

}

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

printf("\n%d",\*(p+i));//In place of \*(p+i) we can write p[i](treating it as ID array)

}

return 0;

}

1. **WAP to read and display elements of 1D array using Dynamic memory allocation(using calloc() function)**

#include<stdio.h>

#include<stdlib.h>

int main( )

{

int \*p,n,i;

printf("Enter the number of blocks we want to reserve:") ;

scanf("%d",&n) ;

p=(int\*)calloc(n,sizeof(int));//malloc() returns void\* so we need to typecast with the specific data type

if(p==NULL)

{

printf("Memory not available\n");

exit(1);

}

else

{

printf("\n Memory allocation successful");

printf ("\nEnter integer values: ") ;

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

{

scanf("%d",p+i);

}

printf("\n Entered values are:");

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

printf("\n%d",\*(p+i));

return 0;

}

}

1. **WAP to perform reallocation of dynamic memory(use of realloc())**

#include<stdio.h>

#include<stdlib.h>

int main()

{

int \*ptr,n,m,i;

printf("\n Enter initial value of n:");

scanf("%d",&n);

ptr=(int \*)calloc(n,sizeof(int));

if(ptr==NULL)

{

printf("\n Memory allocation failure(calloc())");

exit(1);

}

else

{

printf("\n Memory allocation successful");

printf("\n Enter values as per initial requirement:");

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

{

scanf("%d",ptr+i);

}

printf("\n Entered values are:");

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

{

printf("\n%d",\*(ptr+i));

}

m=n;

printf("\n Enter new value of n for reallocation:");

scanf("%d",&n);

ptr=(int \*)realloc(ptr,n\*sizeof(int));

if(ptr==NULL)

{

printf("\n Memory allocation failure while realloation");

exit(2);

}

else

{

printf("\n Memory reallocated successfully");

printf("\n Enter new values as per requirement");

for(i=m;i<n;i++)

{

scanf("%d",ptr+i);

}

printf("\n All values entered are(old+new):");

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

{

printf("\n%d",\*(ptr+i));

}

}

free(ptr);

printf("\n Memory deallocated");

}

return 0;

}

1. **WAP to perform sum of array elements using DMA**

#include<stdio.h>

#include<stdlib.h>

int main( )

{

int \*p,n,i,sum=0;

printf ("Enter the number of integers to be entered ") ;

scanf("%d" ,&n) ;

p=(int\*)malloc(n\*sizeof(int));//malloc() returns void\* so we need to typecast with the specific data type

if(p==NULL)

{

printf("Memory not available\n");

exit(1);

}

else

{

printf("\n Memory allocation was successful");

printf ("\nEnter integer values:") ;

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

{

scanf("%d",p+i);//In place of p+i we can write &p[i](treating it as ID array)

}

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

{

sum=sum+\*(p+i);

}

printf("\n Sum is:%d",sum);

free(p);

printf("\n Memory deallocated");

}

return 0;

}

1. **WAP to implement linear search using DMA**

#include<stdio.h>

#include<stdlib.h>

int main()

{

int \*ptr,i,n,element,loc=-1;

printf("\n Enter no. of values you want to work with:");

scanf("%d",&n);

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

if(ptr==NULL)

{

printf("\n Memory allocation failure");

exit(1);

}

else

{

printf("\n Enter values of elements:");

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

{

scanf("%d",ptr+i);

}

printf("\n Enter the element to search:");

scanf("%d",&element);

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

{

if(\*(ptr+i)==element)

{

loc=i;

break;

}

}

if(loc==-1)

{

printf("\n Element is not found");

}

else

{

printf("\n Element found at location:%d",loc+1);

}

free(ptr);

printf("\n Memory deallocated");

}

return 0;

}

1. **WAP to implement bubble sort using DMA**

#include<stdio.h>

#include<stdlib.h>

int main()

{

int \*ptr,pass,hold;

int n,i;

printf("\n Enter number of values we want to work with:");

scanf("%d",&n);

ptr=(int \*)calloc(n,sizeof(int));

if(ptr==NULL)

{

printf("\n Memory allocation failure...");

exit(1);

}

else

{

printf("\n Memory allocation successful");

printf("\n Enter values:");

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

{

scanf("%d",ptr+i);

}

printf("\n Array before sorting:");

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

{

printf("\n%d",\*(ptr+i));

}

for (pass=1;pass<=n-1;pass++ )

{

// loop to control number of comparisons per pass(There is one comparison less)

for (i=0;i<n-1;i++)

{

// compare adjacent elements and swap them if first

// element is greater than second element

if (\*(ptr+i)>\*(ptr+i+1))

{

hold=\*(ptr+i);

\*(ptr+i)=\*(ptr+i+1);

\*(ptr+i+1)=hold;

} // end if

} // end inner for

} //

printf("\n Array in ascending order is:");

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

{

printf("\n%d",\*(ptr+i));

}

free(ptr);

printf("\n Memory deallocated");

}

return 0;

}

1. **WAP to find area of circle using DMA**

#include<stdio.h>

#include<stdlib.h>

int main()

{

float \*pr,\*pa;

pr=(float\*)malloc(1\*sizeof(float));

pa=(float\*)malloc(1\*sizeof(float));

printf("\n Enter radius value:");

scanf("%f",pr);

\*pa=3.14\*(\*pr)\*(\*pr);

printf("\n Area of the circle is:%f",\*pa);

free(pr);

free(pa);

printf("\n Memory deallocated");

return 0;

}

1. **WAP to find greatest of two numbers using DMA**

#include<stdio.h>

#include<stdlib.h>

int main()

{

int \*ptr;

ptr=(int \*)malloc(2\*sizeof(int));

if(ptr==NULL)

{

printf("\n Mmemory allocation failure");

exit(1);

}

else

{

printf("\n Enter two values:");

scanf("%d %d",ptr+0,ptr+1);

if(\*(ptr+0)==\*(ptr+1))

{

printf("\n Elements are same");

}

else if(\*(ptr+0)>\*(ptr+1))

{

printf("\n Greatest element is:%d",\*(ptr+0));

}

else

{

printf("\n Greatest element is:%d",\*(ptr+1));

}

free(ptr);

printf("\n Memory deallocated");

}

return 0;

}

1. **WAP to demonstrate memory leak situation during DMA**

#include<stdio.h>

int main()

{

int \*p;

p=(int\*)malloc(1\*sizeof(int));

\*p=6;

printf("%d",\*p);

//Memory was not deallocated, hence memory leak may arise

//Solution

//free(ptr);

return 0;

}

**Practice questions:**

* WAP to find the sum and average of double array elements using DMA
* WAP to perform insertion and deletion inside array using DMA
* WAP to count total number of even and odd elements in array using DMA
* WAP to swap the values of two variables(with temporary variable) using DMA
* WAP to implement binary search using DMA